# UNIVERSIDAD CARLOS III DE MADRID

## DEPARTAMENTO DE INFORMÁTICA

### GRADO EN INGENIERÍA INFORMÁTICA


# TRABAJO DE FIN DE GRADO

# Desarrollo de una aplicación generadora y lectora de códigos QR seguros en Android

Autor: Carlos Pliego García

Tutor: Antonio Berlanga de Jesús

Colmenarejo, 20 de Junio de 2013

# Agradecimientos

Antes de nada, he de agradecer a mis padres su paciencia conmigo durante toda mi vida. Mención especial a mi madre, que me ha ayudado a transformar mis ideas de diseño en interfaces para la aplicación y quien, a voz de "que te pilla el toro", ha conseguido que desarrolle el proyecto a tiempo.

A mi hermana Marta, por todos los chistes que nunca logró entender a tiempo.

A mi hermana Piru, por las competiciones por ver quién podía poner la música más alta.

A mi hermano Javi, por sus infatigables conversaciones sobre fútbol.

A mi hermano Salva, por acompañarme al piano cuando tenía el cerebro exhausto.

A mi tutor Antonio, por nuestras discusiones a la hora de elegir el proyecto y por sus consejos y soporte, los cuáles, han sido clave para realizar algo que me parece "un muy buen trabajo".

A mis perras Nana y Pecas, por sus incansables carreras por los pasillos y lametones en la mano.

A Gumpus, por ayudarme a convertir mis composiciones en música.

A los frikis Sergio, Quique, Iñaki y Nico. Sin ellos, las clases habrían sido más productivas pero mucho menos divertidas. A Sergio por su infinitos detalles y su inestimable ayuda en aquellos momentos en los que le necesitaba. A Quique por sus sesiones de música japonesa que aún resuenan en mi cabeza, por todos esos gustos extraños que me ha pegado, y por su muy larga barba. A Iñaki por sus adjetivos inventados como "mej" y "crui", y por esa música "podrida" que tanto nos gusta. A Nico por su cabeza y por esas prácticas que entregábamos a un minuto del límite.

A mi profesora Inmaculada Puebla, por sus elogios hacia mi manera de presentar los trabajos.

Y, por último, a mi abuelo Carlos por haber creído en mí y haberme enseñado a Don McLean, cuya música me ha acompañado de la mano durante todo el desarrollo del proyecto.

# Resumen

En este proyecto se propone el desarrollo de una aplicación para dispositivos móviles Android que permita generar y leer códigos de barras bidimensionales QR cuya información esté cifrada de forma que sólo pueda acceder a ella aquella persona a quién va dirigida. Además estos códigos podrán enviarse a otras aplicaciones instaladas en el dispositivo móvil.

La gran mayoría de aplicaciones de mensajería no cuentan con las medidas de seguridad y privacidad necesarias que posibiliten a los usuarios intercambiar información de carácter sensible sin sufrir un cierto riesgo.

El sistema que se plantea está cualificado para generar y leer códigos QR simples que cualquier usuario con cualquier otra aplicación destinada a ello puede leer. Además, también puede generar códigos a partir de un mensaje que se ha cifrado previamente. De esta forma, la información incluida en ellos sólo podrá ser leída si se descifra antes.

Cuando se decide cifrar la información, la aplicación utiliza los algoritmos criptográficos más seguros. Esta decisión hace de la aplicación una de las mejores opciones cuando se quiere tratar con información muy sensible.

Este sistema puede suponer un gran empujón en el uso de los códigos QR; ya que, a pesar de ser muy utilizados, otorgarles seguridad puede ser algo muy revolucionario. Con esta aplicación se pueden generar códigos que estén dirigidos únicamente a ciertas personas.

Además, una carencia de la gran mayoría de los programas que tratan códigos QR para dispositivos móviles es que sólo permiten leerlos y no generarlos. Incluso, sólo permiten leerlos a través de la cámara, mientras que esta aplicación permite leer los códigos que se encuentren en imágenes almacenadas en la memoria del dispositivo móvil.

Como el sistema permite, además, enviar los códigos generados a más aplicaciones instaladas en el terminal, es muy fácil distribuirlos utilizando aplicaciones de naturaleza de red social como Twitter o Facebook, o aplicaciones de mensajería instantánea como Whatsapp o Line. Incluso permite utilizar servicios de almacenamiento en la nube como Google Drive o Dropbox para guardar los códigos.

**Palabras clave:** Código de barras, QR, almacenamiento en la nube, mensajería instantánea, Android, seguridad, criptografía.

# Abstract

This project proposes the development of an application for Android mobile devices that allows to generate and to read bidimensional QR barcodes whose information is ciphered so it only could be readed by the intended person. Furthermore, these codes could be sent to other applications installed on the mobile device.

Mostly of the messaging applications don't have the necessary security and privacy measures to enable users to exchange sensitive information without suffering some risk.

The system that arises is qualified to generate and read simple QR codes that any user with any other application intended for that purpose may read. Additionally, you can generate codes from a message that was previously ciphered. In this way, the information included in them can only be read if it's decrypted before.

When you choose to encrypt the information, the application uses the most secured cryptographic algorithms. This decision makes the application one of the best options when you want to deal with highly sensitive information.

This system can be a big breakthrough in the use of QR codes; because, despite being widely used, security grant can be very revolutionary. With this application you can generate codes that are intended only for certain people.

Additionally, a lack of the majority of programs form mobile devices that deal with QR codes is that they only allow reading and don't generating them. Even, they just allow to read the codes through the built-in camera while this application allow to read the codes that are in images stored on the memory of the mobile device.

As the system also allows sending the generated codes to more applications installed on the terminal, you can easily distribute them using social network applications like Twitter or Facebook, or instant messaging applications such as Whatsapp or Line. Even allows you to use storage services in the cloud like Google Drive or Dropbox to store the codes.

**Keywords:** Barcode, QR, cloud storage, instant messaging, Android, security, cryptography.

# ÍNDICE

# ÍNDICE DE ILUSTRACIONES

# ÍNDICE DE TABLAS

# 1 INTRODUCCIÓN

## 1.1 CONTEXTO ACTUAL Y OBJETIVO

Desde que el hombre tiene uso de razón, se han utilizado los símbolos para exteriorizar pensamientos o ideas e, incluso, como medio de comunicación con culturas prealfabetizadas. La principal ventaja de los símbolos es que transmiten un mensaje o un concepto al receptor de forma rápida debido a su simplicidad estructural que, únicamente, está definida por elemento visuales.

La ciencia, la tecnología y, sobretodo, el paso de los años, han permitido que se desarrollen sistemas que pueden interpretar estos símbolos y clasificar el objeto al que representan de forma automática. Debido a que estos sistemas no precisan de atención humana, los símbolos que interpretan, en la mayoría de los casos, no representan exactamente el objeto al que identifican. Un claro ejemplo son los códigos de barras, que almacenan información del producto sin representar su figura.

Existen muchos tipos de códigos de barras pero el más famoso es el EAN-128 ya que es el usado para la logística y paquetería y es capaz de representar los 128 caracteres de la tabla ASCII. Aun así, el EAN-128 tiene limitación de espacio y, por eso, hoy en día se está usando cada vez más el código QR ya que, al ser un tipo de código de barras matricial (de dos dimensiones), puede contener mucha más información sobre el producto al que representa. [25]

A día de hoy, existen multitud de aplicaciones de terminales móviles que son capaces de escanear y descifrar estos códigos para brindar al usuario la información que contienen, ya sea un mensaje o un enlace a Internet pero, a pesar de estar cifrados, todos los dispositivos son capaces de descifrarlos y obtener la información sin ningún tipo de privacidad.

Es aquí donde surge la pregunta que se han intentado solventar en el presente proyecto: ¿Sería posible generar un código QR que sólo pueda ser descifrado por las personas o los terminales a quien va dirigida la información que contiene?

# 2 ESTADO DEL ARTE

Para comprender el porqué de las decisiones tomadas en el desarrollo del presente Trabajo de Fin de Grado es esencial conocer y entender los conceptos base que se han tenido en cuenta antes de comenzar con el diseño del proyecto.

## 2.1 SMARTPHONE

Los smartphones o teléfonos inteligentes son teléfonos móviles con características más avanzadas, similares a las de un ordenador. La gran mayoría son táctiles y pueden conectarse a Internet. Para ello utilizan tecnologías como 3G, Wi-Fi, WAP, GPRS, EDGE, HSD, HSDPA, etc.

El desarrollo de estas tecnologías para smartphones ha permitido que las personas se comuniquen unos con otros de forma completamente diferente a como se hacía antes y, en la mayoría de los casos, de forma gratuita. Esto es gracias a las redes sociales y a las aplicaciones de mensajería instantánea. Esta comunicación se produce en tiempo real, lo cual permite que los usuarios se interconecten a través de sus smartphones independientemente del sistema operativo o del desarrollador.

## 2.2 ANDROID

Android es un sistema operativo basado en Linux que ha sido diseñado especialmente para terminales móviles con pantalla táctil. Comprado por Google en 2005, es el principal producto de la Open Handset Alliance. En el año 2011 pasó a ser, con un 50,9% de cuota de mercado, el sistema operativo para terminales móviles más vendido mundialmente y, aun a día de hoy, mantiene su liderazgo.

Debido a la filosofía Open-Source que comparten Linux y Google, el sistema operativo Android tiene una gran comunidad de desarrolladores que extienden la funcionalidad de sus dispositivos escribiendo aplicaciones. Estas aplicaciones, dos tercios de las cuales son gratuitas, se almacenan en el repositorio oficial de Google llamado Google Play. Google realiza varias comprobaciones de seguridad a la aplicación para revisar que no pertenece a la categoría de software malicioso y, una vez la verifica, la hace visible para todos los dispositivos soportados y permite su descarga. [26]

La estructura de este sistema operativo está compuesta por varias aplicaciones que se ejecutan en un framework Java sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas están escritas en lenguaje C e incluyen:

- Surface Manager: Administrador de interfaz gráfica.
- Media Framework: Framewrok OpenCore.
- SQLite: Base de datos relacional basada en SQLite.
- OpenGL ES: Interfaz de programación de API gráfica.
- FreeType: Fuentes en Bitmap y renderizado vectorial.
- WebKit: Motor de renderizado web.
- SGL: Motor de gráficos 2D.
- SSL: Servicios de encriptación Secure Socket Layer.

- libc: Una derivación de las librerías BSD de C estándar, adaptados para dispositivos embebidos basaos en Linux.

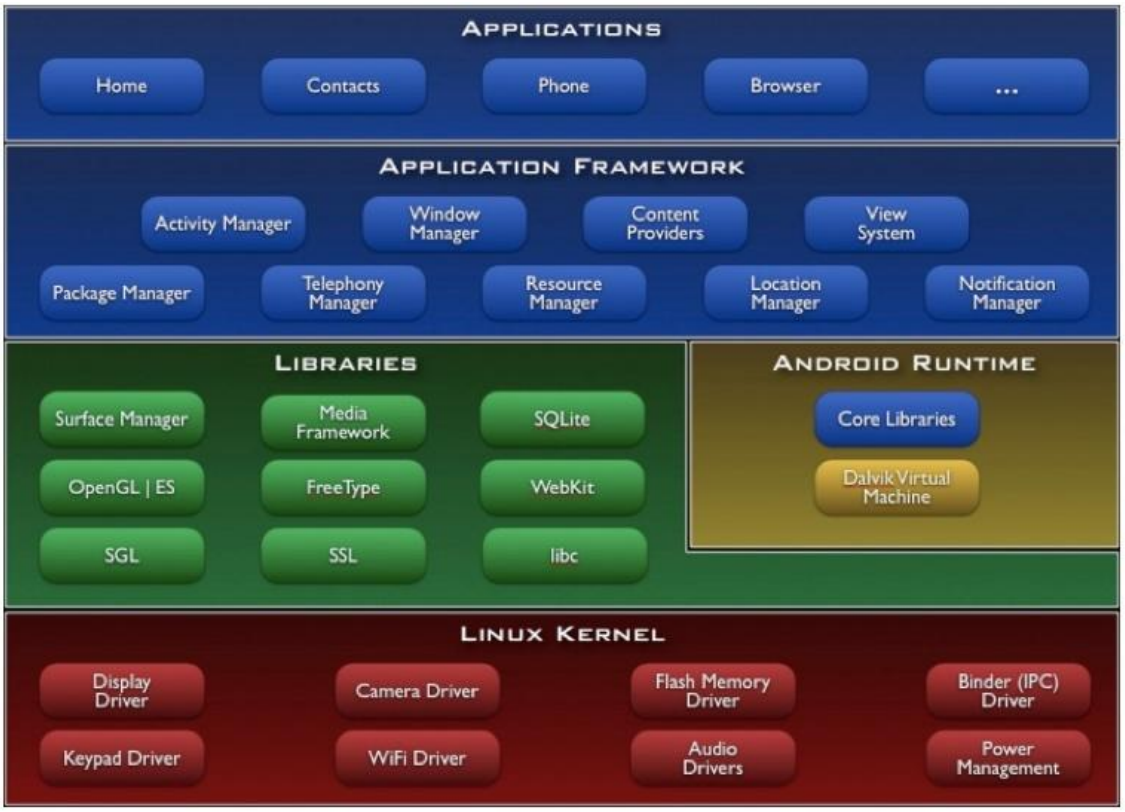



Ilustración 1: Estructura del SO Android

El núcleo del sistema operativo está programado en lenguaje C (más de 2.8 millones de líneas de código), mientras que las librerías están en C y C++ (1.7 millones de líneas). Sin embargo, la interfaz principal y las aplicaciones son programadas en lenguaje Java (2.1 millones de líneas). Es por esto último por lo que existen tantos desarrolladores programando para esta plataforma, ya que el lenguaje Java es versátil, flexible, y fácil. [1]

Android siempre ha apostado por la compatibilidad con muchos y diferentes tipos de smartphones. Esto ha sido una de las razones principales de su gran cantidad de ventas. Además, cada poco tiempo, Google publica nuevas versiones de su sistema operativo que arreglan fallos y añaden nuevas funcionalidades que ayudan al desarrollador a programar aplicaciones más profesionales con más facilidad.

Las versiones de Android hasta la fecha han sido: Apple Pie (v.1.0), Banana Bread (v.1.1), Cupcake (v.1.5), Donut (v.1.6), Éclair (v.2.0), Froyo (v.2.2), Gingerbread (v.2.3), Honeycomb (v.3.0), Ice Cream Sandwich (v.4.0), Jelly Bean (v.4.1). [3]

La versión más utilizada es la Gingerbread (v.2.3) pero las librerías de Android de códigos QR son muy limitadas para esta versión, por lo que se ha decidido utilizar las librerías de la versión Honeycomb (v.3.0) ya que, la siguiente versión más utilizada es la última (la Jelly Bean) pero, como las librerías son escalables (las librerías de la versión 4.0 ya contienen las de la versión 3.0 pero no al revés), se utiliza las de esta versión para obtener una compatibilidad mayor.

A pesar de las ventajas para el usuario y para la empresa que vende el smartphone con este sistema operativo, existe un gran problema para el desarrollador. Este problema consiste en que, al existir tanta variedad de smartphones que utilizan Android, las aplicaciones que se desarrollan tienen que ser muy versátiles para conseguir una máxima compatibilidad y, con las diferencias de resolución de las pantallas, las potencias de procesador tan dispares y las versiones tan distintas de Android, es una tarea verdaderamente ardua es muy difícil conseguir esta compatibilidad.

## 2.3 CÓDIGO QR

Los códigos QR (Quick Response en inglés y respuesta rápida en español) son códigos de barras bidimensionales con la capacidad de almacenar gran cantidad de información codificada dentro de un cuadrado. Son fácilmente reconocibles debido a sus tres cuadrados ubicados en las esquinas (uno en cada esquina superior y otro en esquina inferior izquierda) del cuadrado principal, que permiten detectar la posición del código que se va a leer.

Estos códigos fueron creados por la compañía Denso Wave, subsidiaria de Toyota, en 1994. Lo que se pretendía era que el código permitiera que su contenido se pudiera leer a alta velocidad, ya que su primer uso fue registrar repuestos en el área de fabricación de vehículos. Estos códigos empezaron siendo muy comunes en Japón y, a día de hoy, son muy frecuentes en el mundo entero. [2]

Ilustración 2: Ejemplo de Código QR

Actualmente, con la popularización de los smartphones, la lectura de estos códigos bidimensionales QR se ha vuelto muy sencilla, ya que sólo consiste en enfocar el

código con la cámara, por lo que el uso de este código se ha incrementado drásticamente y está siendo utilizado por las empresas con fines publicitarios.

Los principales usos que se les da a los códigos QR son:

- **Publicidad:** El mensaje puede ser una URL de una empresa o información sobre un producto suyo.
- **Papelería corporativa:** Puede contener la información de una tarjeta de visita sin la necesidad de imprimirla.
- **Gestión de stocks y almacenes:** Puede contener toda la información necesaria de un artículo ordenado en una estantería, indicando si éste se está agotando para realizar un nuevo pedido o si, por el contrario, existen demasiados y es recomendable promocionarlos por medio de ofertas.
- **Concursos:** Se puede hacer intervenir al instante a un consumidor en una competición, llevándole, incluso, a una página web donde puede inscribirse para ganar premios.
- **Asesoramiento al consumidor:** Puede proporcionar una información adicional del producto con el que viene acompañado.

Para descifrar un código QR solamente se necesita un dispositivo con cámara de fotos y un lector compatible. Pero ahora, gracias a los smartphones, gran parte de la población cumple con estos requisitos. De ahí que su popularidad haya crecido tanto últimamente.

En la web, existen varias aplicaciones que permiten generar los códigos QR con el mensaje que se desee pero en el repositorio de aplicaciones de Google existen muy pocas. Por otro lado, en este repositorio hay muchas aplicaciones que escanean códigos QR y los descifran desde la cámara pero no existen las que los escaneen y descifren directamente desde un archivo de imagen.

Estas dos carencias se van a intentar subsanar con la aplicación que se va a desarrollar.

## 2.4 SEGURIDAD

### 2.4.1 CONCEPTO GENERAL

Del latín *securitas*. Se puede referir a la ausencia de riesgo o a la confianza en algo o alguien. [27] Relativo a la informática, la seguridad está concebida para proteger la infraestructura computacional y a los usuarios. En el entorno que nos interesa, la seguridad se anticipa y protege al usuario de que alguien sepa información suya a través de los códigos QR.

Algunos tipos de amenazas que pueden darse por una mala seguridad son los programas maliciosos, los intrusos o la manipulación malintencionada del terminal (como un *overclock* exagerado por parte de alguien que no es propietario del smartphone).

Existen varias formas de evitar que alguna de estas amenazas se den en un terminal; como, por ejemplo, la codificación de la información, los escudos de red y de aplicaciones, los antivirus o los sistemas de respaldo remotos.

La seguridad que se va a intentar incluir en la aplicación no consiste en la inmunidad del smartphone, sino en la protección de los datos que contiene un código QR. Como se ha dicho anteriormente, los códigos QR no ofrecen ningún tipo de protección frente a una persona que no debería leerlo ya que la información que contiene no está cifrada. Por eso, se ha decidido utilizar la técnica de la codificación para cifrar el mensaje. De esta forma, una persona que no esté autorizada a conocer la información que éste contiene sólo obtendría, al escanearlo, una cadena alfanumérica sin ningún sentido.

### 2.4.2 CONCEPTOS IMPORTANTES

Para entender más adelante los términos referentes a la seguridad sin ningún problema, se van a explicar los conceptos con los que se va a estar tratando durante todo el desarrollo del proyecto.

- **Encriptación:** La encriptación es el proceso de cambiar los datos de forma que sólo puedan ser leídos por el receptor al que va destinado. [28]
- **Algoritmo de cifrado AES:** La información que se incluya en el código QR tiene que ser cifrada para que una persona ajena no pueda conocerla. Para ello, se utilizará el algoritmo de cifrado AES que comprenderá la sección principal del proceso de cifrado del texto del código. AES es un algoritmo de cifrado que no está roto (es decir, es seguro) y que está considerado suficientemente seguro como para proteger la información altamente secreta. Este algoritmo utiliza una clave de cifrado de tamaño variable desde 128 bits hasta 256, pasando por 192 bits. Cuanto más grande sea el tamaño de la clave, más complejidad tendrá el algoritmo de cifrado. La descripción de AES consiste en dos partes, la primera es el proceso de cifrado y la segunda es el proceso de generación de las subclaves, una aproximación muy simple se muestra en la siguiente figura: [29]

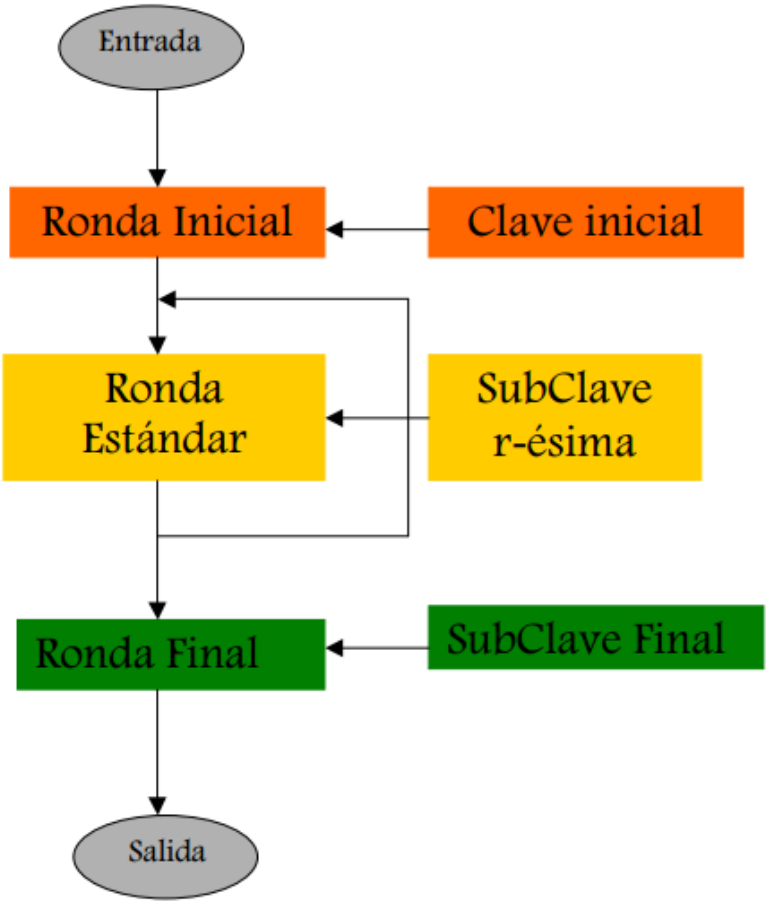



Ilustración 3: Diagrama del algoritmo de cifrado AES

- **Salt:** En el contexto de seguridad, un *salt* es un conjunto de bits aleatorios que se concatenan a la contraseña para generar una nueva clave y evitar los ataques de diccionario.
- **Vector de inicialización:** Cuando se va a realizar un cifrado por bloques como el que se va a realizar en esta aplicación, es aconsejable utilizar un bloque de bits aleatorio llamado vector de inicialización que permite generar resultados distintos del cifrado utilizándose la misma clave y el mismo texto.
- **Algoritmo de codificación *Base64*:** Este algoritmo permite representar datos binarios en formato ASCII. Para ello se utiliza un alfabeto compuesto por 65 caracteres pertenecientes al US-ASCII. La siguiente ilustración muestra el alfabeto del algoritmo Base64: [30]

The Base 64 Alphabet

| Value | Encoding | Value | Encoding | Value | Encoding | Value | Encoding |
|---|---|---|---|---|---|---|---|
| 0 | A | 17 | R | 34 | i | 51 | z |
| 1 | B | 18 | S | 35 | j | 52 | 0 |
| 2 | C | 19 | T | 36 | k | 53 | 1 |
| 3 | D | 20 | U | 37 | l | 54 | 2 |
| 4 | E | 21 | V | 38 | m | 55 | 3 |
| 5 | F | 22 | W | 39 | n | 56 | 4 |
| 6 | G | 23 | X | 40 | o | 57 | 5 |
| 7 | H | 24 | Y | 41 | p | 58 | 6 |
| 8 | I | 25 | Z | 42 | q | 59 | 7 |
| 9 | J | 26 | a | 43 | r | 60 | 8 |
| 10 | K | 27 | b | 44 | s | 61 | 9 |
| 11 | L | 28 | c | 45 | t | 62 | + |
| 12 | M | 29 | d | 46 | u | 63 | / |
| 13 | N | 30 | e | 47 | v | | |
| 14 | O | 31 | f | 48 | w | (pad) | = |
| 15 | P | 32 | g | 49 | x | | |
| 16 | Q | 33 | h | 50 | y | | |

Ilustración 4: Alfabeto utilizado en el algoritmo Base64

# 3 MARCO REGULADOR

Antes de comenzar a analizar la aplicación, se han estudiado las leyes vigentes y las normativas técnicas que atañen al proyecto. Seguidamente, se procederá a definir cada una de estas leyes.

## 3.1 LEY GENERAL DE LAS TELECOMUNICACIONES

La ley general de las telecomunicaciones fundó un régimen plenamente liberalizado en la asistencia de servicios y el establecimiento y explotación de redes de telecomunicaciones, descubriendo el sector a la libre competencia entre operadores. [18]

Su objetivo no es otro que el de asegurar y avalar el secreto de las comunicaciones.

El artículo que interesa de esta ley es el 36 de la Ley General de Telecomunicaciones sobre el "Cifrado en las redes y servicios de comunicaciones electrónicas". Este artículo pertenece a la ley datada del 3 de Noviembre de 2003 y dice así: [18]

1. "**Cualquier tipo de información** que se transmita por redes de comunicaciones electrónicas **podrá ser protegida mediante procedimientos de cifrado**."
2. "El cifrado es un instrumento de seguridad de la información. Entre sus condiciones de uso, cuando se utilice para proteger la confidencialidad de la información, **se podrá imponer la obligación de facilitar a un órgano de la Administración General del Estado o a un organismo público, los algoritmos o cualquier procedimiento de cifrado utilizado, así como la obligación de facilitar sin coste alguno los aparatos de cifra a efectos de su control** de acuerdo con la normativa vigente."

El primero de los puntos señala que no se incumple ninguna norma y está permitido el cifrar todo tipo de información que se vaya a transmitir por redes de comunicaciones electrónicas. La aplicación que se va a desarrollar va a emplear técnicas de cifrado sobre los datos que, más tarde, pueden enviarse mediante aplicaciones de terceros por redes de comunicaciones electrónicas. De esta forma, se puede concluir que la aplicación asegura el secreto de las comunicaciones.

El segundo punto informa de que la Administración General del Estado o un organismo público pueden requerir los algoritmos o procedimientos empleados y los aparatos de cifra. Como se tiene pensado publicar la aplicación en el repositorio oficial de Google (Google Play), cualquier organismo puede descargar el software desde allí y ponerse en contacto con su creador mediante la dirección de correo electrónico que es obligatorio definir cuando se publica algún software.

## 3.2 LEY ORGÁNICA DE PROTECCIÓN DE DATOS

La ley orgánica de protección de datos tiene por objeto **garantizar y proteger**, en lo que concierne al tratamiento de **los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas**, **y** especialmente de **su honor e intimidad personal y familiar**. [19]

Esta Ley Orgánica, datada del 13 de Diciembre, manifiesta en el artículo 9 llamado "Seguridad de los datos" que:

"**El responsable del fichero, y**, en su caso, **el encargado del tratamiento deberán adoptar las medidas de índole técnica y organizativas necesarias que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado**, habida cuenta del estado de la tecnología, la naturaleza de los datos almacenados y los riesgos a que están expuestos, ya provengan de la acción humana o del medio físico o natural." [19]

Este punto señala que, tanto el usuario de la aplicación como el desarrollador de ella, deben tomar medidas de seguridad para que no se vea afectada la privacidad de los datos. Para respetar y cumplir con esta ley, se tomarán las siguientes decisiones:

- Los algoritmos de cifrado que se empleen deberán ser muy seguros, de modo que ningún ataque contra la privacidad de los datos sea viable.
- La información secreta contenida en un código QRypt sólo será accesible por el autor del mensaje y por los participantes que éste considere con tal derecho.
- Ningún sistema externo al dispositivo móvil desde donde se realice el cifrado almacenará datos sobre el usuario. La aplicación se diseñará de tal forma que los códigos que se generen a partir de ella, se envíen desde aplicaciones de terceros. De esta forma, la seguridad de los datos contenidos en el código a la hora de enviarlo no recaerá sobre la aplicación que se va a desarrollar; sino, sobre el usuario y la aplicación que escoja para enviar los datos.

# 4 ANÁLISIS

El objetivo de esta sección no es otro que el de obtener una especificación detallada del sistema de información que se va a desarrollar. Tras esta especificación, que establecerá la base del sistema, se detallará el diseño de éste. El análisis tratará de comprender las necesidades de las que se precisa una resolución y conformará el problema.

## 4.1 DEFINICIÓN DEL SISTEMA

Antes de profundizar en la captación y el análisis de requisitos es necesario detallar su funcionamiento, especificar los problemas que debe resolver, determinar las restricciones con las que tendrá que enfrentarse y definir el entrono operacional que va a requerir.

### 4.1.1 ALCANCE DEL SISTEMA

El sistema de información que se pretende desarrollar consiste en una aplicación que va a posibilitar a los usuarios de smartphones que utilicen el sistema operativo Android la generación y el futuro escaneo de códigos QR cifrados. Además, la aplicación permitirá compartir estos códigos con cualquier persona mediante el programa de mensajería móvil que se desee.

La aplicación debe ser capaz de generar y leer códigos QR. Además, deberá poder cifrar un mensaje de forma rápida pero robusta y, posteriormente, transcribirlo a un código QR. Este código se denominará QRypt. Después, debe adaptar este código a una imagen para que pueda ser compartida mediante una aplicación externa de mensajería móvil o, en su defecto, guardada en la memoria SD del teléfono. Finalmente, debe ser capaz de escanear una imagen que contenga un código QR, aislarlo de los demás elementos que compongan la imagen, y descifrar el mensaje original con ayuda de una contraseña. El sistema debe implementar algoritmos criptográficos para el cifrado del mensaje original.

En el caso de generar un código QRypt, el sistema dará la opción de cifrar el mensaje mediante los siguientes métodos:

- **Contraseña:** El mensaje se cifrará con una contraseña que se haya acordado entre el emisor y el receptor como clave.
- **Imei:** El mensaje se cifrará con el imei del terminal que genera el código como clave. De esta forma, el mensaje sólo podrá recuperarlo el terminal que generó el código.
- **Modelo:** El mensaje se cifrará con el modelo del terminal que genera el código como clave. De esta forma, el mensaje sólo podrá recuperarlo un terminal que comparta el modelo del terminal que generó el código.
- **Fecha:** El mensaje se cifrará con una fecha que introducirá el emisor como clave. Si la fecha es futura, el mensaje no se podrá leer. Las comprobaciones se harán con la fecha de Internet.
- **Patrón:** El mensaje se cifrará con un patrón semejante al que utiliza Android en el desbloqueo de sus terminales como clave.

Más adelante se definirá la funcionalidad completa de la aplicación.

### 4.1.2 RESTRICCIONES GENERALES

El equipo de desarrollo y la normativa legal especificada en el marco regulador han impuesto las siguiente restricciones para el presente proyecto:

- La interfaz de la aplicación debe ser sencilla, usable, accesible y amigable.
- Es necesario que la aplicación esté programada en lenguaje de programación Java.
- La aplicación debe ser compatible con el mayor número posible de versiones del sistema operativo Android.
- La aplicación debe ser capaz de cumplir a la perfección con todas las funcionalidades que se hallan en los requisitos de usuario.
- Los requisitos de usuario coincidirán con los que se especificaron en las reuniones que se han mantenido con el tutor y se complementarán con las decisiones que el alumno haya considerado pertinentes.
- Los algoritmos criptográficos que se utilicen en el sistema deben ser seguros.
- La información personal de los usuarios de la aplicación y los datos que se generen o traten quedarán únicamente en posesión del usuario. La aplicación no enviará datos anónimamente a ningún servidor, bien en posesión del desarrollador, bien en posesión de cualquier otra entidad.

### 4.1.3 ENTORNO OPERACIONAL

La aplicación que se va a desarrollar necesita que el terminal en el que se va a ejecutar cuente con unos requisitos mínimos para asegurar que el sistema funciona correctamente. Estos requisitos son:

- Una versión del sistema operativo Android igual o superior a la Honeycomb (v.3.0). Esta versión se beneficia de una API con la que no cuentan las anteriores versiones y es esencial para soportar un diseño estándar de Google llamado Holo que mejora notablemente la interacción usuario-aplicación.
- Una cantidad de memoria RAM no menor a 700MB, ya que el terminal debe ser capaz de tratar con rapidez los datos y las herramientas que utiliza la aplicación. Además, al ser necesaria también una versión del sistema operativo Android que utiliza la memoria RAM con mucha asiduidad, se necesita una cantidad de memoria que, a pesar de ser la normal en dispositivos que soportan este sistema operativo, no es baja.

## 4.2 ENTORNO DE DESARROLLO

En esta sección se van a exponer todos los datos y la información pertinente sobre los elementos hardware y software que componen el entorno de desarrollo con el que se le dará vida a la aplicación.

### 4.2.1 EQUIPOS

Se utilizarán los siguientes dispositivos físicos para diseñar, desarrollar y probar la aplicación:

- **Dispositivo móvil Samsung Galaxy SII**
  - Modelo: GT-I9100.
  - Versión del sistema operativo: 4.1.2 Jelly Bean (API 16). [3]
  - Procesador: Dual-Core 1.2 GHz Cortex-A9. [4]
  - RAM: 1GB.
  - Versión de banda base: I9100XXMS2
  - Versión del Kernel: 3.0.31 - 1069234 dpi@DELL237 #3 SMP PREEMPT
  - Número de compilación: JZO54K.I9100XWLSN
- **Ordenador de sobremesa hecho a piezas**
  - Procesador: Intel Core i7-2600K 3.4GHz.
  - Sistema operativo: Windows 8 Professional 64 bits.
  - Memoria RAM: 8GB DDR3.
  - Capacidad del disco duro: 1TB.
- **Ordenador portátil Asus**
  - Modelo: A55A-SX465H.
  - Procesador: Intel Core i7-3630QM 2.4GHz.
  - Sistema operativo: Windows 8 Professional 64 bits.
  - Memoria RAM: 8GB DDR3.
  - Capacidad del disco duro: 500GB.

### 4.2.2 LENGUAJE DE PROGRAMACIÓN

Debido a que no hay que realizar ninguna tarea en la consola de comandos de Windows, el único lenguaje que se utilizará durante el desarrollo de la aplicación será el lenguaje de programación Java ya que éste es el lenguaje que se emplea en la programación de las aplicaciones para el sistema operativo Android. En este caso, se utilizará la API de Android 11 ya que es la mínima que proporciona las librerías básicas necesarias para un diseño Holo.

### 4.2.3 ENTORNO DE DESARROLLO

El entorno de desarrollo que se empleará para llevar a cabo el proyecto será el programa Eclipse. Este entorno está programado en Java y cuenta con un gran número de complementos para el desarrollo y diseño de aplicaciones para el sistema operativo Android.

Se ha decidido escoger este entorno de desarrollo ya que es uno de los más usado para programar aplicaciones para el sistema operativo Android, cuenta con un emulador virtual de dispositivo Android en el que se puede comprobar la funcionalidad de la aplicación en diferentes tamaños de pantalla y versiones del

sistema operativo, y es el entorno de programación que más se ha utilizado durante la carrera.

La versión de Eclipse, el nombre de los complementos y el número de versión del entorno de desarrollo software para Android (llamado SDK, "Software Development Kit") serán los siguientes:

- Versión de Eclipse: 3.4.
- Complementos: Android Development Tools (ADT).
- Android SDK: 3.0 (API 11).

## 4.3 REQUISITOS DE USUARIO

En circunstancias normales, la función de los requisitos de usuario no sería otra que la de recoger la información explícita sobre qué es lo que quiere el cliente y qué es lo que necesita. Como este caso varía de la regla general ya que se trata de un proyecto de fin de grado, estos requisitos se obtendrán en sesiones de trabajo entre el tutor del proyecto y el alumno encargado de realizarlo. El fin de estas reuniones es numerar y definir los procesos y tareas que el sistema de información (en este caso, la aplicación) debería poder llevar a cabo sin ningún problema, y, por otro lado, las limitaciones que éste debería tener.

Debido a la naturaleza de los requisitos de usuario, se ha decidido dividir su especificación en **requisitos de capacidad** del sistema y **requisitos de restricción** sobre el sistema.

Cada requisito deberá estar definido en una tabla. Las tablas se compondrán de los siguientes atributos:

- **Identificador:** Es preciso que cada requisito de usuario esté vinculado a un identificador exclusivo y unívoco para que su seguimiento futuro pueda realizarse de forma más simple. El identificador estará formado por dos elementos clave:
    - **Siglas:** Se referirán al tipo de requisito de usuario que sea. Si es un requisito de usuario de capacidad, serán *RUC*; mientras que, si es un requisito de usuario de restricción, serán *RUR*.
    - **Número:** El número de requisito que sea. Se representará siempre con 2 cifras.

  De esta forma, un ejemplo de identificador de requisito de usuario de capacidad podría ser el "*RUC-08*", que equivaldría al *requisito de usuario de capacidad número 8*.

- **Nombre:** El nombre que se le da al requisito. Debe ser un poco descriptivo y, no necesariamente, unívoco.
- **Descripción:** La descripción de lo que debería poder hacer el sistema o, por el contrario, lo que no debería. Esta especificación no debería ser extensa y debería explicar de forma correcta en qué consiste en requisito.
- **Prioridad:** La prioridad de un requisito frente a otros. Los requisitos de más prioridad deberán implementarse antes que los de menor prioridad en el proceso de diseño o implementación. Los valores que puede tomar este atributo son: *Alta, Media y Baja*.
- **Necesidad:** La importancia de que un requisito se implemente o no. Los valores que puede recibir este campo son:
    - **Primario:** Es un requisito de usuario obligatorio que debe ser implementado.
    - **Secundario:** Es un requisito de usuario que debería ser implementado pero, al contrario que el anterior, no es obligatorio.

  - **Opcional:** Es un requisito de usuario que se puede dejar de implementar.
- **Estabilidad:** Mide lo estable que será un requisito de usuario en relación a los cambios que se puedan producir en el sistema. Los requisitos podrán ser de uno de estos dos tipos:
  - **Alta:** El requisito no se modificará durante la vida del sistema de información.
  - **Baja:** El requisito puede modificarse puntualmente.
- **Fuente:** Especifica cuál es el origen del requisito de usuario. Puede tomar dos valores:
  - **Alumno:** Si ha sido el alumno el que ha propuesto el requisito de usuario.
  - **Tutor:** Si, por el contrario, el tutor ha sido quien lo ha propuesto.

En la Tabla 1 se expone una plantilla que representa a un requisito de usuario general.

| IDENTIFICADOR: RU@-XX | |
|---|---|
| **Nombre:** | |
| **Descripción:** | |
| **Prioridad:** | ☐ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☐ Baja |
| **Fuente:** | ☐ Alumno ☐ Tutor |

Tabla 1: Plantilla de requisito de usuario

### 4.3.1 REQUISITOS DE CAPACIDAD

Los requisitos de usuario de capacidad especifican las tareas y funciones que debe cumplir el sistema de información para resolver un problema o, en su defecto, alcanzar un objetivo. Los requisitos de usuario de capacidad recogidos en las sesiones de trabajo con el tutor son:

| IDENTIFICADOR: RUC-01 | |
|---|---|
| **Nombre:** | Modo generar QR |
| **Descripción:** | El usuario será capaz de acceder al modo generar QR desde la pantalla principal de la aplicación. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐ Alumno ☑Tutor |

Tabla 2: RUC-01

| IDENTIFICADOR: RUC-02 | |
|---|---|
| **Nombre:** | Modo generar QRypt |
| **Descripción:** | El usuario será capaz de acceder al modo generar QRypt desde la pantalla principal de la aplicación. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐ Alumno ☑Tutor |

Tabla 3: RUC-02

| IDENTIFICADOR: RUC-03 | |
|---|---|
| **Nombre:** | Modo leer por cámara |
| **Descripción:** | El usuario será capaz de acceder al modo leer por cámara desde la pantalla principal de la aplicación. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐ Alumno ☑Tutor |

Tabla 4: RUC-03

| IDENTIFICADOR: RUC-04 | |
|---|---|
| **Nombre:** | Modo leer desde fichero |
| **Descripción:** | El usuario será capaz de acceder al modo leer desde fichero desde la pantalla principal de la aplicación. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐ Alumno ☑Tutor |

Tabla 5: RUC-04

| IDENTIFICADOR: RUC-05 | |
|---|---|
| **Nombre:** | Escoger imagen para leer |
| **Descripción:** | El usuario será capaz de escoger una imagen de la galería para descifrar el código QR o QRypt que contenga, si es que contiene alguno de los dos. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐ Alumno ☑Tutor |

Tabla 6: RUC-05

| IDENTIFICADOR: RUC-06 | |
|---|---|
| **Nombre:** | Escribir mensaje para cifrar |
| **Descripción:** | El usuario será capaz de escribir un mensaje para poder cifrarlo o para poder generar un código QR simple. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐ Alumno ☑Tutor |

Tabla 7: RUC-06

| IDENTIFICADOR: RUC-07 | |
|---|---|
| **Nombre:** | Escribir contraseña para cifrar |
| **Descripción:** | El usuario será capaz de escribir una contraseña con la que cifrar el mensaje para generar un código QRypt. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐ Alumno ☑Tutor |

Tabla 8: RUC-07

| IDENTIFICADOR: RUC-08 | |
|---|---|
| **Nombre:** | Permitir cifrar con imei |
| **Descripción:** | El usuario será capaz de cifrar un mensaje usando el imei del terminal como contraseña para generar un código QRypt. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 9: RUC-08

| IDENTIFICADOR: RUC-09 | |
|---|---|
| **Nombre:** | Permitir cifrar con modelo |
| **Descripción:** | El usuario será capaz de cifrar un mensaje usando el modelo del terminal como contraseña para generar un código QRypt. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 10: RUC-09

| IDENTIFICADOR: RUC-10 | |
|---|---|
| **Nombre:** | Permitir cifrar con la fecha |
| **Descripción:** | El usuario será capaz de cifrar un mensaje usando una fecha como contraseña para generar un código QRypt. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 11: RUC-10

| IDENTIFICADOR: RUC-11 | |
|---|---|
| **Nombre:** | Permitir cifrar con un patrón |
| **Descripción:** | El usuario será capaz de cifrar un mensaje usando patrón semejante a los de desbloqueo como contraseña para generar un código QRypt. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☑ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 12: RUC-11

| IDENTIFICADOR: RUC-12 | |
|---|---|
| **Nombre:** | Contador de caracteres |
| **Descripción:** | El usuario será capaz de ver el número de caracteres del mensaje que está escribiendo en tiempo real |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☑ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 13: RUC-12

| IDENTIFICADOR: RUC-13 | |
|---|---|
| **Nombre:** | Escoger color del código |
| **Descripción:** | El usuario será capaz de escoger el color que quiera para el código QRypt. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 14: RUC-13

| IDENTIFICADOR: RUC-14 | |
|---|---|
| **Nombre:** | Escoger modo de transmisión |
| **Descripción:** | El usuario será capaz de escoger la aplicación a la que quiere enviar el código para compartirlo. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 15: RUC-14

| IDENTIFICADOR: RUC-15 | |
|---|---|
| **Nombre:** | Carpeta de salida |
| **Descripción:** | El usuario será capaz de elegir la carpeta en la que se guardarán las imágenes que contengan los códigos QR y QRypt generados. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 16: RUC-15

| IDENTIFICADOR: RUC-16 | |
|---|---|
| **Nombre:** | Panel de ajustes |
| **Descripción:** | El usuario será capaz de cambiar los ajustes por defecto del cifrado. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 17: RUC-16

| IDENTIFICADOR: RUC-17 | |
|---|---|
| **Nombre:** | Novedades de la versión |
| **Descripción:** | El usuario será capaz de comprobar las novedades que se han producido de una versión a otra. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 18: RUC-17

| IDENTIFICADOR: RUC-18 | |
|---|---|
| **Nombre:** | Escribir contraseña para descifrar |
| **Descripción:** | El usuario será capaz de escribir una contraseña para descifrar un código QRypt si éste ha sido cifrado con este método. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐Alumno ☑Tutor |

Tabla 19: RUC-18

| IDENTIFICADOR: RUC-19 | |
|---|---|
| **Nombre:** | Mostrar mensaje descifrado |
| **Descripción:** | El usuario será capaz de ver el mensaje descifrado. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐Alumno ☑Tutor |

Tabla 20: RUC-19

| IDENTIFICADOR: RUC-20 | |
|---|---|
| **Nombre:** | Abrir URLs |
| **Descripción:** | El usuario será capaz de decidir si quiere ser redirigido cuando el mensaje descifrado sea una URL. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 21: RUC-20

| IDENTIFICADOR: RUC-21 | |
|---|---|
| **Nombre:** | Navegación sencilla |
| **Descripción:** | La interfaz deberá ser sencilla e intuitiva. Se obligará al usuario a seguir los pasos que se propongan como si fuera un asistente. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 22: RUC-21

| IDENTIFICADOR: RUC-22 | |
|---|---|
| **Nombre:** | Diseño para daltónicos |
| **Descripción:** | El diseño de la interfaz deberá ser perfectamente accesible para gente con problemas de daltonismo. Se evitará el uso de tonos verdes y rojos. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 23: RUC-22

| IDENTIFICADOR: RUC-23 | |
|---|---|
| **Nombre:** | Menú compartir |
| **Descripción:** | El usuario será capaz de elegir la aplicación desde el menú "compartir" para imágenes de Android. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 24: RUC-23

| IDENTIFICADOR: RUC-24 | |
|---|---|
| **Nombre:** | Escoger modo de cifrado |
| **Descripción:** | El usuario será capaz elegir el modo de cifrado que quiere para su mensaje si es que ha elegido generar un código QRypt. Podrá elegir entre contraseña, imei, modelo, fecha y patrón. Además, se podrá elegir más de una opción. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 25: RUC-24

| IDENTIFICADOR: RUC-25 | |
|---|---|
| **Nombre:** | Panel de control deslizable |
| **Descripción:** | El usuario será capaz de acceder a un panel de control deslizable para cambiar los ajustes desde cualquier ventana de la aplicación. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 26: RUC:25

| IDENTIFICADOR: RUC-26 | |
|---|---|
| **Nombre:** | Volver atrás |
| **Descripción:** | El usuario será capaz de volver a la primera actividad de la aplicación desde cualquier otra actividad únicamente pulsando un botón. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☑ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 27: RUC-26

### 4.3.2 REQUISITOS DE RESTRICCIÓN

Los requisitos de usuario de restricción especifican todo tipo de limitación en la forma de llevar a cabo las funciones que debería realizar la aplicación. Además, estos requisitos detallan cómo resolver el problema o cómo se debe alcanzar el objetivo. Los requisitos de usuario de restricción recogidos en las sesiones de trabajo son:

| IDENTIFICADOR: RUR-01 | |
|---|---|
| **Nombre:** | Longitud mensaje |
| **Descripción:** | La longitud máxima del mensaje que se va a cifrar será de 2000 caracteres. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 28: RUR-01

| IDENTIFICADOR: RUR-02 | |
|---|---|
| **Nombre:** | No mostrar contraseña |
| **Descripción:** | En el caso de que haya que introducir una contraseña, ésta será ocultada con asteriscos. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐Alumno ☑Tutor |

Tabla 29: RUR-02

| IDENTIFICADOR: RUR-03 | |
|---|---|
| **Nombre:** | Contraseña para descifrar |
| **Descripción:** | La contraseña que se introducirá en el proceso de descifrado del mensaje deberá ser exactamente igual a la que se usó para cifrarlo. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐Alumno ☑Tutor |

Tabla 30: RUR-03

| IDENTIFICADOR: RUR-04 | |
|---|---|
| **Nombre:** | Seguridad óptima |
| **Descripción:** | Se utilizarán algoritmos de cifrado seguros. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐Alumno ☑Tutor |

Tabla 31: RUR-04

| IDENTIFICADOR: RUR-05 | |
|---|---|
| **Nombre:** | Dependencia de Internet |
| **Descripción:** | La aplicación no necesitará una conexión a Internet para cifrar y descifrar un código QR o un código QRypt salvo que el método de cifrado escogido sea por fecha. También se necesitará para comprobar actualizaciones y para compartir un código. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☑ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☐Alumno ☑Tutor |

Tabla 32: RUR-05

| IDENTIFICADOR: RUR-06 | |
|---|---|
| **Nombre:** | Disponibilidad de idiomas |
| **Descripción:** | La aplicación deberá estar traducida y localizada al castellano y al inglés. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☑ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 33: RUR-06

| IDENTIFICADOR: RUR-07 | |
|---|---|
| **Nombre:** | Tiempo de cifrado |
| **Descripción:** | La aplicación deberá tardar un máximo de 30 segundos en generar un código QR y un máximo de 1 minuto en generar un código QRypt. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☑ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 34: RUR-07

| IDENTIFICADOR: RUR-08 | |
|---|---|
| **Nombre:** | Color negro para QR |
| **Descripción:** | La aplicación no permitirá elegir ningún color para un código QR simple. El negro está reservado para estos códigos. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 35: RUR-08

| IDENTIFICADOR: RUR-09 | |
|---|---|
| **Nombre:** | Color negro para QRypt |
| **Descripción:** | La aplicación debe permitir elegir el color en el que se quiere cifrar el código QRypt. El negro no estará permitido. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 36: RUR-09

| IDENTIFICADOR: RUR-10 | |
|---|---|
| **Nombre:** | Introducir contraseña |
| **Descripción:** | La aplicación no debe permitir introducir la contraseña cuando el código QRypt que se va a leer haya sido cifrado mediante imei, modelo o fecha. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Fuente:** | ☑Alumno ☐Tutor |

Tabla 37: RUR-10

## 4.4 CASOS DE USO

En la presente sección se especificarán los casos de uso del sistema. Estos casos de uso se obtienen de los requisitos de usuario que se han definido en el apartado anterior. El objetivo de los casos de uso es describir las posibles acciones que el usuario puede realizar sobre el sistema. Cada caso de uso puede satisfacer uno o varios requisitos de usuario, en especial aquellos que estén directamente involucrados con la actividad que va a resolver el caso de uso.

El orden en el que se va a proceder para realizar esta sección es el siguiente:

1. Se van a ilustran los casos de uso que se consideran principales en un diagrama general sencillo.
2. Se van a construir diagramas adicionales más detallados que contienen todos y cado uno de los casos de uso que se han especificado en los requisitos. La función de este paso no es otra que la de seguir un proceso de análisis incremental.
3. Se completará con una descripción detallada de cada caso de uso que aparezca en los diagramas.

Al igual que se ha hecho en la sección de requisitos de usuario, se va a especificar cada caso de uso con una tabla que contendrá unos atributos. Estos atributos serán:

- **Identificador:** Es preciso que cada caso de uso esté vinculado a un identificador exclusivo y unívoco para que su seguimiento futuro pueda realizarse de forma más simple. El identificador estará formado por dos elementos clave: unas siglas que indicarán que se trata de un caso de uso, y un número de dos cifras que indicará el caso de uso que es. Las siglas reservadas para definir un caso de uso son "CU". De esta forma, un ejemplo de identificador de caso de uso podría ser "*CU-13*", que equivaldría al *caso de uso número 13*.
- **Nombre:** El nombre resumirá la función del caso de uso que se esté definiendo.
- **Actores:** Define qué actor o actores interactúan con el caso de uso.
- **Descripción:** Describe cómo un actor debería interactuar con la aplicación y la respuesta que el sistema ofrecería.
- **Precondiciones:** Define el estado del sistema necesario para la realización del caso de uso.
- **Postcondiciones:** Define el estado del sistema tras la realización del caso de uso.
- **Secuencia principal:** Establece el orden de las acciones que son necesarias para alcanzar el caso de uso.
- **Secuencia alternativa:** En el caso de que exista, describe caminos de naturaleza diferente a la secuencia principal para alcanzar el caso de uso.
- **Requisitos:** Indica los requisitos de usuario con los que esté relacionado el caso de uso.

En la Tabla 39 se expone una plantilla que representa a un caso de uso general.

| IDENTIFICADOR: CU-XX | |
|---|---|
| **Nombre:** | |
| **Actores:** | |
| **Descripción:** | |
| **Precondiciones:** | |
| **Postcondiciones:** | |
| **Secuencia principal:** | |
| **Secuencia alternativa:** | |
| **Requisitos:** | |

Tabla 38: Plantilla de caso de uso

### 4.4.1 CASO DE USO GENERAL

En la Ilustración 3 se muestra el diagrama del caso de uso general para la aplicación.

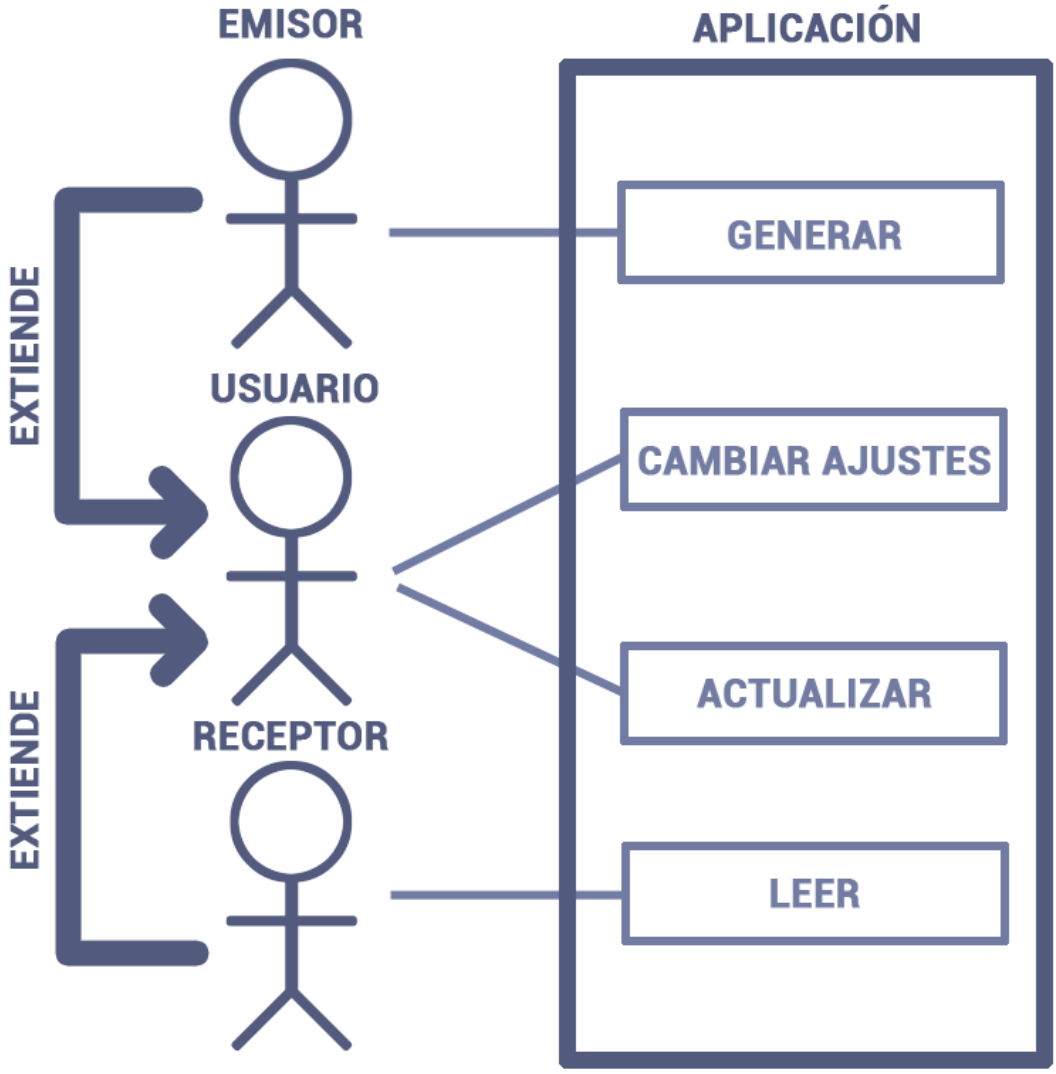



Ilustración 5: Caso de uso principal

Como se puede observar en la Ilustración 5, hay 3 posibles casos de uso generales: generar un código QR o QRypt, cambiar los ajustes de la aplicación, actualizar la aplicación y extraer el mensaje de un código QR o QRypt.

En la Ilustración 5 existen 3 actores pero 2 de ellos son derivaciones del usuario principal. El papel de cada uno de estos actores es el siguiente:

- **Usuario:** Es el usuario general. Aquel que va a hacer uso de la aplicación. Debido a la gran diferenciación entre el módulo de generación y el de lectura, resulta conveniente derivar de este actor otros dos. Aun así, el usuario principal, tenga el rol que tenga, podrá cambiar los ajustes de la aplicación y ver las novedades de la versión que ejecuta.
- **Emisor:** Es el usuario cuyo objetivo es el de generar un código QR con un mensaje. El mensaje puede estar cifrado o no. En el primero de los casos, el emisor está generando un código QRypt mientras que, en el segundo sólo genera un código QR simple. Este actor deriva del usuario principal.
- **Receptor:** Es el usuario cuyo objetivo no es otro que el de recuperar un mensaje (cifrado o no) que se halla en un código QR. Este actor deriva del usuario principal.

Como más adelante se va a desglosar con profundidad los casos de uso "Generar" y "Leer", a continuación se va a explicar únicamente el acceso a éstos y los dos casos de uso que puede realizar el usuario.

| **IDENTIFICADOR: CU-01** | |
|---|---|
| **Nombre:** | Generar. |
| **Actores:** | Emisor. |
| **Descripción:** | El emisor selecciona la opción de generar un QR o un QRypt desde la pantalla principal. |
| **Precondiciones:** | Ejecutar la aplicación. |
| **Postcondiciones:** | Se accede al módulo de generación de QR o de QRypt. |
| **Secuencia principal:** | 1. Ejecutar aplicación.<br>2. Seleccionar la opción generar QR o la opción generar QRypt. |
| **Secuencia alternativa:** | |
| **Requisitos:** | RUC-01, RUC-02. |

Tabla 39: CU-01

| IDENTIFICADOR: CU-02 | |
|---|---|
| **Nombre:** | Leer. |
| **Actores:** | Receptor. |
| **Descripción:** | El receptor selecciona la opción de leer un código QR desde la pantalla principal. |
| **Precondiciones:** | Ejecutar la aplicación. |
| **Postcondiciones:** | Se accede al módulo de lectura de código QR mediante la cámara o mediante un fichero. |
| **Secuencia principal:** | 1. Ejecutar aplicación.<br>2. Seleccionar la opción leer QR por cámara o por fichero.. |
| **Secuencia alternativa:** | (1). El receptor accede al módulo de lectura del código QR desde el menú "Abrir con..." de las imágenes de Android. |
| **Requisitos:** | RUC-03, RUC-04. |

Tabla 40: CU-02

| IDENTIFICADOR: CU-03 | |
|---|---|
| **Nombre:** | Cambiar ajustes. |
| **Actores:** | Usuario. |
| **Descripción:** | El usuario abre el panel deslizable y cambia los ajustes de la aplicación desde cualquier pantalla. |
| **Precondiciones:** | Ejecutar la aplicación. |
| **Postcondiciones:** | Se modifica el valor de los ajustes por defecto de la aplicación. |
| **Secuencia principal:** | 1. Ejecutar aplicación.<br>2. Abrir el menú deslizable.<br>3. Seleccionar cualquier opción de ajustes.<br>4. Modificar los ajustes. |
| **Secuencia alternativa:** | (1). Ejecutar la aplicación y abrir cualquier módulo de ella. |
| **Requisitos:** | RUC-15, RUC-16. |

Tabla 41: CU-03

| IDENTIFICADOR: CU-04 | |
|---|---|
| **Nombre:** | Actualizar. |
| **Actores:** | Usuario. |
| **Descripción:** | El usuario abre el panel deslizable y selecciona la opción de ver las novedades de la versión. |
| **Precondiciones:** | Ejecutar la aplicación. |
| **Postcondiciones:** | Diálogo con las novedades de la versión actual. |
| **Secuencia principal:** | 1. Ejecutar aplicación.<br>2. Abrir el menú deslizable.<br>3. Seleccionar la opción de ver las novedades de la versión. |
| **Secuencia alternativa:** | |
| **Requisitos:** | RUC-17. |

Tabla 42: CU-04

### 4.4.2 CASOS DE USO DEL MÓDULO GENERAR

En esta sección se va a profundizar en los casos de uso relacionados con la generación de códigos QR o QRypt.

En la Ilustración 4 se muestra un diagrama más completo en el que se desglosan los casos de uso para el módulo de generación de códigos QR o QRypt.

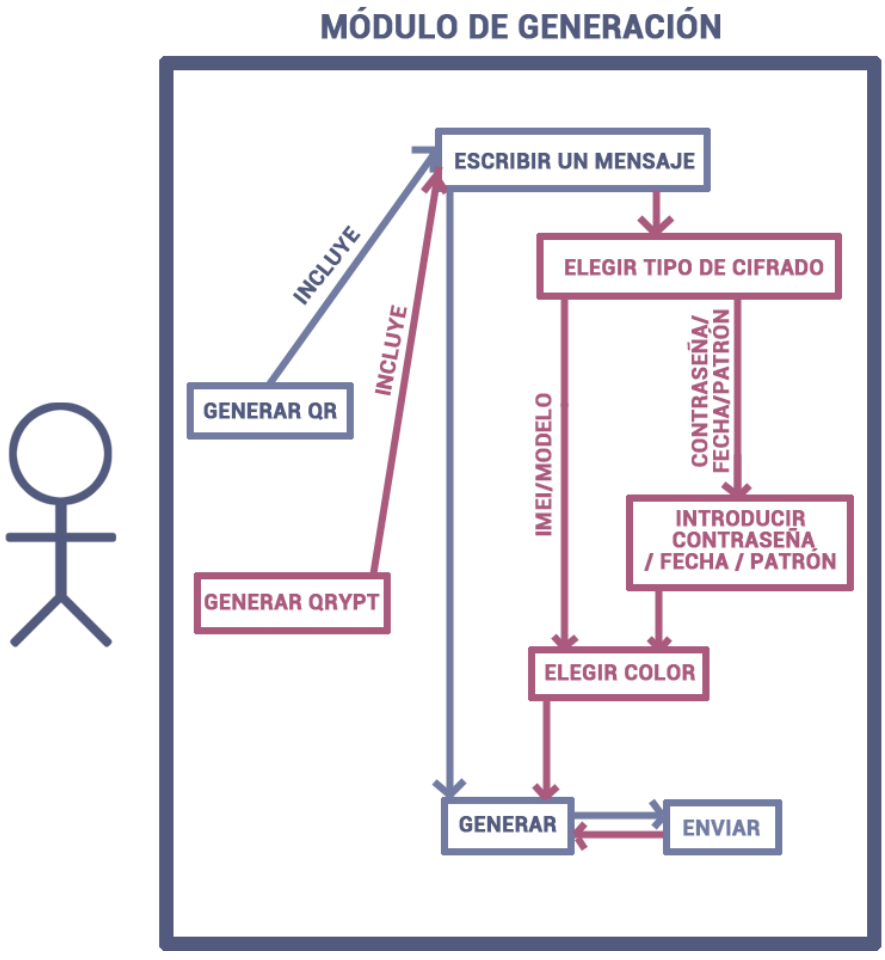



Ilustración 6: Casos de uso para el módulo de generación

Como se puede observar, en este caso sólo hay un actor. Este actor es el emisor, cuyo objetivo es introducir un mensaje en un código QR. El actor puede elegir entre cifrar este mensaje o no, siendo la primera de las opciones la que le ofrece más acciones. Se escoja la opción que se escoja, la aplicación dará oportunidad de compartir el código.

Se puede ver que el proceso de generación está formado por distintos casos de uso y que cada uno de ellos actúa como una acción cuya resolución es necesaria para alcanzar el objetivo.

Existe un caso de uso que es condicional (sólo se ejecuta cuando se cumple una condición). Este caso de uso es "Introducir contraseña/fecha/patrón" ya que, si no se ha elegido uno de esos tipos de cifrado, no se va a ejecutar.

A continuación, se exponen los casos de uso que se han mostrado en la Ilustración 6 siguiendo el formato que impone la plantilla de la Tabla 38.

| **IDENTIFICADOR: CU-05** | |
|---|---|
| **Nombre:** | Escribir un mensaje. |
| **Actores:** | Emisor. |
| **Descripción:** | El emisor escribe el mensaje que quiere incluir en un código QR. |
| **Precondiciones:** | Acceder al modo de generación de QR o de QRypt. |
| **Postcondiciones:** | Se escribe el mensaje. |
| **Secuencia principal:** | 1. Acceder al modo de generación de QR.<br>2. Escribir el mensaje. |
| **Secuencia alternativa:** | (1). Acceder al modo de generación de QRypt.<br>(2). Escribir el mensaje. |
| **Requisitos:** | RUC-06, RUC-12, RUC-21, RUC-22, RUC-25, RUR-01. |

Tabla 43: CU-05

| **IDENTIFICADOR: CU-06** | |
|---|---|
| **Nombre:** | Elegir tipo de cifrado. |
| **Actores:** | Emisor. |
| **Descripción:** | El emisor elige el tipo de cifrado con el que quiere cifrar el mensaje. |
| **Precondiciones:** | Escribir un mensaje para que sea cifrado. |
| **Postcondiciones:** | Se elige la forma de cifrado. |
| **Secuencia principal:** | 1. Acceder al modo de generación de QRypt.<br>2. Escribir el mensaje.<br>3. Elegir modo de cifrado |
| **Secuencia alternativa:** | |
| **Requisitos:** | RUC-08, RUC-09, RUC-10, RUC-11, RUC-21, RUC-22, RUC-24, RUC-25, RUC-26. |

Tabla 44: CU-06

| IDENTIFICADOR: CU-07 | |
|---|---|
| **Nombre:** | Introducir contraseña/fecha/patrón. |
| **Actores:** | Emisor. |
| **Descripción:** | El emisor introduce la contraseña que se le pide para cifrar el mensaje. |
| **Precondiciones:** | Haber elegido el modo de cifrado por contraseña, por patrón o por fecha.. |
| **Postcondiciones:** | Se introduce la contraseña. |
| **Secuencia principal:** | 1. Acceder al modo de generación de QRypt.<br>2. Escribir el mensaje.<br>3. Elegir modo de cifrado (contraseña/fecha/patrón).<br>4. Introducir la contraseña para cifrar. |
| **Secuencia alternativa:** | |
| **Requisitos:** | RUC-07, RUC-21, RUC-22, RUC-25, RUC-26, RUR-02. |

Tabla 45: CU-07

| IDENTIFICADOR: CU-08 | |
|---|---|
| **Nombre:** | Elegir color. |
| **Actores:** | Emisor. |
| **Descripción:** | El emisor elige el color con el que quiere que se muestre el código QRypt. |
| **Precondiciones:** | Haber elegido el modo de cifrado e introducido la contraseña en caso de que se solicitase. |
| **Postcondiciones:** | Se elige el color. |
| **Secuencia principal:** | 1. Acceder al modo de generación de QRypt.<br>2. Escribir el mensaje.<br>3. Elegir modo de cifrado (contraseña/fecha/patrón).<br>4. Introducir la contraseña para cifrar<br>5. Elegir color. |
| **Secuencia alternativa:** | (1). Acceder al modo de generación de QRypt.<br>(2). Escribir el mensaje.<br>(3). Elegir modo de cifrado (Imei/Modelo).<br>(4). Elegir color. |
| **Requisitos:** | RUC-13, RUC-21, RUC-22, RUC-25, RUC-26, RUR-08, RUR-09. |

Tabla 46: CU-08

| IDENTIFICADOR: CU-09 | |
|---|---|
| **Nombre:** | Generar. |
| **Actores:** | Emisor. |
| **Descripción:** | La aplicación genera el código según las opciones introducidas en los pasos anteriores. |
| **Precondiciones:** | Acceder al modo de generación de código QR o al modo de generación de código QRypt. |
| **Postcondiciones:** | Mensaje introducido en un código QR. o mensaje cifrado introducido en un código QRypt. |
| **Secuencia principal:** | 1. Acceder al modo de generación de QR.<br>2. Escribir el mensaje.<br>3. Presionar el botón "Aceptar". |
| **Secuencia alternativa:** | (1). Acceder al modo de generación de QRypt.<br>(2). Escribir el mensaje.<br>(3). Elegir modo de cifrado (contraseña/fecha/patrón).<br>(4). Introducir la contraseña para cifrar<br>(5). Elegir color.<br>(6). Presionar el botón "Aceptar". |
| **Secuencia alternativa 2:** | (1). Acceder al modo de generación de QRypt.<br>(2). Escribir el mensaje.<br>(3). Elegir modo de cifrado (Imei/Modelo).<br>(4). Elegir color.<br>(5). Presionar el botón "Aceptar". |
| **Requisitos:** | RUR-04, RUR-05, RUR-07. |

Tabla 47: CU-09

| IDENTIFICADOR: CU-10 | |
|---|---|
| **Nombre:** | Enviar. |
| **Actores:** | Emisor. |
| **Descripción:** | La aplicación abre el menú compartir de Android para que el usuario envíe el código donde quiera. |
| **Precondiciones:** | Generar un código. |
| **Postcondiciones:** | Código compartido. |
| **Secuencia principal:** | 1. Acceder al modo de generación de QR.<br>2. Escribir el mensaje.<br>3. Presionar el botón "Aceptar".<br>4. Pulsar el botón compartir y seleccionar la aplicación. |
| **Secuencia alternativa:** | (1). Acceder al modo de generación de QRypt.<br>(2). Escribir el mensaje.<br>(3). Elegir modo de cifrado (contraseña/fecha/patrón).<br>(4). Introducir la contraseña para cifrar<br>(5). Elegir color.<br>(6). Presionar el botón "Aceptar".<br>(7). Pulsar el botón compartir y seleccionar la aplicación. |
| **Secuencia alternativa 2:** | (1). Acceder al modo de generación de QRypt.<br>(2). Escribir el mensaje.<br>(3). Elegir modo de cifrado (Imei/Modelo).<br>(4). Elegir color.<br>(5). Presionar el botón "Aceptar".<br>(6). Pulsar el botón compartir y seleccionar la aplicación |
| **Requisitos:** | RUC-14, RUC-21, RUC-22, RUC-25, RUC-26. |

Tabla 48: CU-10

### 4.4.3 CASOS DE USO DEL MÓDULO LEER

En esta sección se va a profundizar en los casos de uso relacionados con la lectura de códigos QR y QRypt.

En la Ilustración 5 se muestra un diagrama más completo en el que se desglosan los casos de uso para el módulo de la lectura de códigos QR y QRypt.

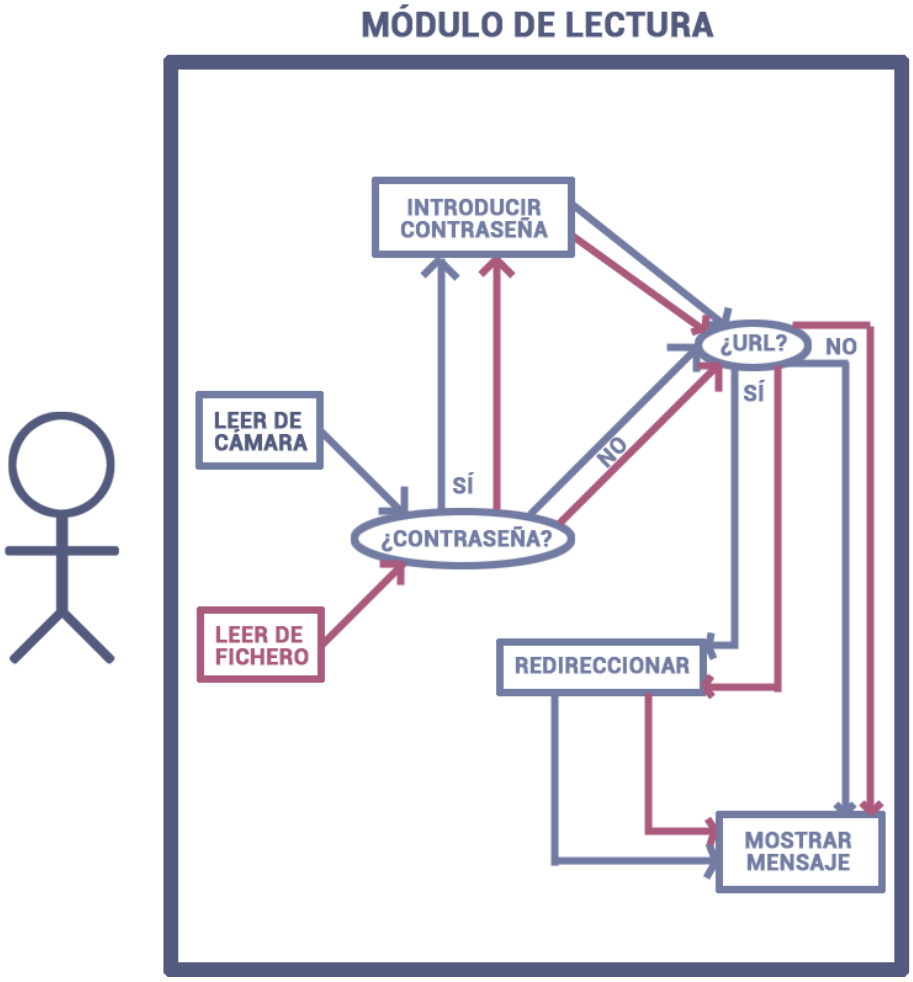



Ilustración 7: Casos de uso para el módulo de lectura

Como se puede observar, en este caso sólo hay un actor. Este actor es el receptor, cuyo objetivo es extraer el mensaje que ha sido introducido en un código QR. El actor puede elegir entre leer el código mediante la cámara del terminal o buscar una imagen en la galería que contenga ese código. Se escoja la opción que se escoja, una vez se ha extraído el mensaje, no hay diferencia en la ejecución de las opciones.

Se puede ver que el proceso de lectura está formado por distintos casos de uso y que cada uno de ellos actúa como una acción cuya resolución es necesaria para alcanzar el objetivo. Al igual que en el módulo de generación, existen unos casos de uso cuya ejecución estará sujeta a una condición. Se han utilizado óvalos para representar estas condiciones.

Sin importar que opción de lectura se elija, la aplicación comprobará si el mensaje ha sido cifrado con una contraseña. Si es así, procederá a pedirle al receptor que introduzca una. Una vez superado este punto, la aplicación comprobará si el mensaje es una URL o no y, de ser así, preguntará al receptor si quiere ser redireccionado a ella y, después, mostrará la URL o el mensaje por pantalla.

A continuación, se exponen los casos de uso que se han mostrado en la Ilustración 7 siguiendo el formato que impone la plantilla de la Tabla 36.

| IDENTIFICADOR: CU-11 | |
|---|---|
| **Nombre:** | Leer de cámara. |
| **Actores:** | Receptor. |
| **Descripción:** | El receptor apunta con la cámara del dispositivo a un código QR o QRypt. |
| **Precondiciones:** | Acceder al modo de lectura por cámara. |
| **Postcondiciones:** | Se obtiene el mensaje (cifrado o no). |
| **Secuencia principal:** | 1. Acceder al modo de lectura por cámara.<br>2. Obtener un mensaje (cifrado o no) |
| **Secuencia alternativa:** | |
| **Requisitos:** | RUC-03, RUC-21, RUC-22, RUC-26. |

Tabla 49: CU-11

| IDENTIFICADOR: CU-12 | |
|---|---|
| **Nombre:** | Leer de fichero. |
| **Actores:** | Receptor. |
| **Descripción:** | El receptor selecciona una imagen que se encuentre en la memoria del dispositivo y que contenga un código QR o QRypt. |
| **Precondiciones:** | Acceder al modo de lectura por fichero. |
| **Postcondiciones:** | Se obtiene el mensaje (cifrado o no). |
| **Secuencia principal:** | 1. Acceder al modo de lectura por fichero.<br>2. Obtener un mensaje (cifrado o no) |
| **Secuencia alternativa:** | |
| **Requisitos:** | RUC-04, RUC-05, RUC-21, RUC-22, RUC-23, RUC-26. |

Tabla 50: CU-12

| IDENTIFICADOR: CU-13 | |
|---|---|
| **Nombre:** | Introducir contraseña. |
| **Actores:** | Receptor. |
| **Descripción:** | El receptor introduce la contraseña para descifrar el código QRypt. |
| **Precondiciones:** | Obtener el mensaje y que éste esté cifrado con una contraseña. |
| **Postcondiciones:** | Contraseña introducida. |
| **Secuencia principal:** | 1. Acceder al modo de lectura por cámara.<br>2. Obtener un mensaje cifrado<br>3. Introducir la contraseña. |
| **Secuencia alternativa:** | (1). Acceder al modo de lectura por fichero.<br>(2). Obtener un mensaje cifrado.<br>(3). Introducir la contraseña. |
| **Requisitos:** | RUC-18, RUC-21, RUC-22, RUC-26, RUR-02, RUR-03 RUR-05, RUR-10. |

Tabla 51: CU-13

| IDENTIFICADOR: CU-14 | |
|---|---|
| **Nombre:** | Redireccionar. |
| **Actores:** | Receptor. |

| | |
|---|---|
| **Descripción:** | El receptor señala si quiere o no que se le redireccione a la URL que se halla en el mensaje del QR o QRypt. |
| **Precondiciones:** | Obtener el mensaje y que éste sea una URL. |
| **Postcondiciones:** | Se redirecciona a la URL. |
| **Secuencia principal:** | 1. Acceder al modo de lectura por cámara.<br>2. Obtener un mensaje y que éste sea una URL.<br>3. Redireccionar a la URL. |
| **Secuencia alternativa:** | (1). Acceder al modo de lectura por fichero.<br>(2). Obtener un mensaje y que éste sea una URL.<br>(3). Redireccionar a la URL. |
| **Secuencia alternativa 2:** | (1). Acceder al modo de lectura por cámara.<br>(2). Obtener un mensaje cifrado y que éste sea una URL.<br>(3). Introducir la contraseña.<br>(4). Redireccionar a la URL. |
| **Secuencia alternativa 3:** | (1). Acceder al modo de lectura por fichero.<br>(2). Obtener un mensaje cifrado y que éste sea una URL.<br>(3). Introducir la contraseña.<br>(4). Redireccionar a la URL. |
| **Requisitos:** | RUC-20, RUC-21, RUC-22, RUC-26. |

Tabla 52: CU-14

| **IDENTIFICADOR: CU-15** | |
|---|---|
| **Nombre:** | Mostrar mensaje. |
| **Actores:** | Receptor. |
| **Descripción:** | Se muestra por pantalla el mensaje introducido en el código QR o QRypt. |
| **Precondiciones:** | Obtener el mensaje y que éste sea una URL. |
| **Postcondiciones:** | Se redirecciona a la URL. |
| **Secuencia principal:** | 1. Acceder al modo de lectura por cámara.<br>2. Obtener un mensaje.<br>3. Mostrar el mensaje. |
| **Secuencia alternativa:** | (1). Acceder al modo de lectura por fichero.<br>(2). Obtener un mensaje.<br>(3). Mostrar el mensaje. |
| **Secuencia alternativa 2:** | (1). Acceder al modo de lectura por cámara.<br>(2). Obtener un mensaje cifrado.<br>(3). Introducir la contraseña.<br>(4). Mostrar el mensaje. |
| **Secuencia alternativa 3:** | (1). Acceder al modo de lectura por |

| | |
|---|---|
| | fichero.<br>(2). Obtener un mensaje cifrado.<br>(3). Introducir la contraseña.<br>(4). Mostrar el mensaje. |
| **Secuencia alternativa 4:** | (1). Acceder al modo de lectura por cámara.<br>(2). Obtener un mensaje cifrado y que éste sea una URL.<br>(3). Introducir la contraseña.<br>(4). Redireccionar a la URL.<br>(5). Mostrar el mensaje. |
| **Secuencia alternativa 5:** | (1). Acceder al modo de lectura por fichero.<br>(2). Obtener un mensaje cifrado.<br>(3). Introducir la contraseña.<br>(4). Redireccionar a la URL.<br>(5). Mostrar el mensaje. |
| **Requisitos:** | RUC-19, RUC-21, RUC-22, RUC-25, RUC-26. |

Tabla 53: CU-15

## 4.5 REQUISITOS DE SOFTWARE

Los requisitos de software se obtienen de los requisitos de usuario y del modelo de casos de uso. Estos requisitos especifican la funcionalidad que la aplicación tendrá; es decir, lo que tiene que hacer el sistema.

Cuando se especifican los requisitos de software, es recomendable separarlos en dos grupos:

- **Requisitos de software funcionales:** Definen lo que tiene que hacer la aplicación y su propósito. Se obtienen de los casos de uso.
- **Requisitos de software no funcionales:** Definen cómo deben realizarse las funcionalidades del sistema de información. Estos requisitos se dividirán en las siguientes clases:
    - **Requisitos de operación:** Especifican cómo debe realizar el sistema de información las tareas.
    - **Requisitos de interfaz:** Especifican la interacción entre el usuario y el sistema y entre los módulos de éste.
    - **Requisitos de rendimiento:** Indican la carga que se espera que tenga que soportar el sistema.
    - **Requisitos de recursos:** Especifican los medios y recursos necesarios para que el sistema funcione.
    - **Requisitos de comprobación:** Indican las limitaciones que afectan a cómo el sistema de información tiene que verificar los datos de entras y salida.
    - **Requisitos de seguridad:** Definen los medios con los que el sistema se protegerá de amenazas en la integridad, confidencialidad y disponibilidad.

Como se ha hecho anteriormente con los requisitos de usuario y los casos de uso, estos requisitos se van a redactar utilizando una plantilla. La plantilla tendrá los siguientes atributos:

- **Identificador:** Es preciso que cada requisito de software esté vinculado a un identificador exclusivo y unívoco para que su seguimiento futuro pueda realizarse de forma más simple. El identificador estará formado por dos elementos clave: unas siglas que indicarán que se trata de un requisito de software, y un número de dos cifras que indicará el requisito de software que es. Las siglas reservadas para definir un requisito de software son "RS". De esta forma, un ejemplo de identificador de requisito de software podría ser "*RS-24*", que equivaldría al *requisito de software número 24*.
- **Descripción:** La descripción de en qué consiste el requisito de software.
- **Prioridad:** La prioridad de un requisito frente a otros. Los requisitos de más prioridad deberán implementarse antes que los de menor prioridad en el proceso de diseño o implementación. Los valores que puede tomar este atributo son: *Alta, Media y Baja*.
- **Necesidad:** La importancia de que un requisito se implemente o no. Los valores que puede recibir este campo son:

  - **Primario:** Es un requisito de software obligatorio que debe ser implementado.
  - **Secundario:** Es un requisito de software que debería ser implementado pero, al contrario que el anterior, no es obligatorio.
  - **Opcional:** Es un requisito de software que se puede dejar de implementar.
- **Estabilidad:** Mide lo estable que será un requisito de usuario en relación a los cambios que se puedan producir en el sistema. Los requisitos podrán ser de uno de estos dos tipos:
  - **Alta:** El requisito no se modificará durante la vida del sistema de información.
  - **Baja:** El requisito puede modificarse puntualmente.
- **Requisitos:** Especifica cuál o cuáles son los requisitos de usuario de los que se extrae el requisito de software.

En la Tabla 54 se expone una plantilla que representa a un requisito de software general.

| IDENTIFICADOR: RS-XX | |
|---|---|
| **Descripción:** | |
| **Prioridad:** | ☐ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☐ Baja |
| **Requisitos:** | |

Tabla 54: Plantilla de requisito de software

### 4.5.1 REQUISITOS FUNCIONALES

Como se ha explicado ya, estos requisitos definen lo que tiene que hacer la aplicación y su propósito. A pesar de que se obtienen de los casos de uso, teniendo en cuenta que éstos últimos se obtienen de los requisitos de usuario, se indicarán los requisitos de usuario de los que surgen estos requisitos de software.

Las requisitos de software funcionales se listan a continuación:

| IDENTIFICADOR: RS-01 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda acceder al modo de generación de códigos QR. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-01. |

Tabla 55: RS-01

| IDENTIFICADOR: RS-02 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda acceder al modo de generación de códigos QRypt. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-02. |

Tabla 56: RS-02

| IDENTIFICADOR: RS-03 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda acceder al modo de lectura por cámara. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-03. |

Tabla 57: RS-03

| IDENTIFICADOR: RS-04 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda acceder al modo de lectura desde fichero. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-04, RUC-23. |

Tabla 58: RS-04

| IDENTIFICADOR: RS-05 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir una imagen de la galería desde el modo de lectura desde fichero. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-05, RUC-23. |

Tabla 59: RS-05

| IDENTIFICADOR: RS-06 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda escribir un mensaje. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-06. |

Tabla 60: RS-06

| IDENTIFICADOR: RS-07 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda escribir un mensaje de hasta 2000 caracteres como máximo. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | RUR-01. |

Tabla 61: RS-07

| IDENTIFICADOR: RS-08 | |
|---|---|
| **Descripción:** | El sistema de información deberá mostrar un contador de caracteres que indique el número de caracteres que se han escrito en el mensaje. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-12. |

Tabla 62: RS-08

| IDENTIFICADOR: RS-09 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario introduzca una contraseña para cifrar el mensaje. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-07. |

Tabla 63: RS-09

| IDENTIFICADOR: RS-10 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir un tipo de cifrado como mínimo. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-24. |

Tabla 64: RS-10

| IDENTIFICADOR: RS-11 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir un color para el código si éste es QRypt. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-13, RUR-08, RUR-09. |

Tabla 65: RS-11

| IDENTIFICADOR: RS-12 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir el modo de cifrado por contraseña. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-07. |

Tabla 66: RS-12

| IDENTIFICADOR: RS-13 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir el modo de cifrado por imei del terminal. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-08. |

Tabla 67: RS-13

| IDENTIFICADOR: RS-14 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir el modo de cifrado por modelo del terminal. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-09. |

Tabla 68: RS-14

| IDENTIFICADOR: RS-15 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir el modo de cifrado por fecha. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-10. |

Tabla 69: RS-15

| IDENTIFICADOR: RS-16 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir el modo de cifrado por patrón. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-11. |

Tabla 70: RS-16

| IDENTIFICADOR: RS-17 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir una aplicación de terceros a la que compartir el código generado. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-14. |

Tabla 71: RS-17

| IDENTIFICADOR: RS-18 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda generar un código QR o QRypt con los datos introducidos. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-01, RUC-02, RUC-06, RUC-07, RUC-07, RUC-08, RUC-09, RUC-10, RUC-11, RUC-13, RUC-24. |

Tabla 72: RS-18

| IDENTIFICADOR: RS-19 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda elegir la carpeta en la que quiere que se guarden los código generados. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-15. |

Tabla 73: RS-19

| IDENTIFICADOR: RS-20 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda cambiar las opciones por defecto. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-16. |

Tabla 74: RS-20

| IDENTIFICADOR: RS-21 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario introduzca una contraseña para descifrar el QRypt. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-18. |

Tabla 75: RS-21

| IDENTIFICADOR: RS-22 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario elija si quiere que se le redireccione a una URL cuando el mensaje contenga una. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-20. |

Tabla 76: RS-22

| IDENTIFICADOR: RS-23 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario extraiga el mensaje de un código a partir de las opciones seleccionadas. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-03, RUC-04, RUC-05, RUC-18, RUC-23, RUR-03, RUR-05, RUR-10. |

Tabla 77: RS-23

| IDENTIFICADOR: RS-24 | |
|---|---|
| **Descripción:** | El sistema de información deberá mostrar el mensaje del código por pantalla una vez se ha descifrado. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-19. |

Tabla 78: RS-24

| IDENTIFICADOR: RS-25 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario pueda ver las novedades de la versión. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-17. |

Tabla 79: RS-25

### 4.5.2 REQUISITOS DE OPERACIÓN

| IDENTIFICADOR: RS-26 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario acceda al modo de generación de códigos QR desde la pantalla principal de la interfaz. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-01. |

Tabla 80: RS-26

| IDENTIFICADOR: RS-27 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario acceda al modo de generación de códigos QRypt desde la pantalla principal de la interfaz. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-02. |

Tabla 81: RS-27

| IDENTIFICADOR: RS-28 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario acceda al modo de lectura de códigos por cámara desde la pantalla principal de la interfaz. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-03. |

Tabla 82: RS-28

| IDENTIFICADOR: RS-29 | |
|---|---|
| **Descripción:** | El sistema de información deberá permitir que el usuario acceda al modo de lectura de códigos por fichero desde la pantalla principal de la interfaz. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-04. |

Tabla 83: RS-29

| IDENTIFICADOR: RS-30 | |
|---|---|
| **Descripción:** | Se mostrará el menú deslizable con los ajustes de la aplicación cuando se pulse el botón "menú" del terminal. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-25. |

Tabla 84: RS-30

| IDENTIFICADOR: RS-31 | |
|---|---|
| **Descripción:** | Se mostrará el menú deslizable con los ajustes de la aplicación cuando se deslice el dedo por el borde izquierdo de la pantalla. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-25. |

Tabla 85: RS-31

| IDENTIFICADOR: RS-32 | |
|---|---|
| **Descripción:** | El sistema de información debe permitir al usuario navegar por la interfaces de los módulos de generación de códigos QR y QRypt para corregir o modificar la información incluida en los campos. Para ello se hará uso del botón “atrás” del dispositivo y de botones para avanzar. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-21, RUC-26. |

Tabla 86: RS-32

| IDENTIFICADOR: RS-33 | |
|---|---|
| **Descripción:** | El sistema de información mostrará un diálogo para escoger los distintos tipos de cifrado:<br>• Contraseña.<br>• Imei.<br>• Modelo.<br>• Fecha.<br>• Patrón. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-24. |

Tabla 87: RS-33

| IDENTIFICADOR: RS-34 | |
|---|---|
| **Descripción:** | El sistema de información mostrará un diálogo para escoger los distintos colores para el QRypt:<br>• Verde.<br>• Rojo.<br>• Azul.<br>• Naranja.<br>• Morado.<br>• Marrón.<br>• Rosa.<br>• Gris.<br>• Random. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-13. |

Tabla 88: RS-34

| IDENTIFICADOR: RS-35 | |
|---|---|
| **Descripción:** | El sistema de información mostrará un botón para seleccionar la aplicación de terceros a la que se desea mandar el código. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-14. |

Tabla 89: RS-35

| IDENTIFICADOR: RS-36 | |
|---|---|
| **Descripción:** | El sistema de información deberá mandar la imagen a la aplicación de terceros que se haya seleccionado. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-14. |

Tabla 90: RS-36

| IDENTIFICADOR: RS-37 | |
|---|---|
| **Descripción:** | El sistema de información mostrará un botón de "Siguiente" en la barra de acción del módulo de generación para generar el código con el mensaje escrito. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | |

Tabla 91: RS-37

| IDENTIFICADOR: RS-38 | |
|---|---|
| **Descripción:** | El sistema de información mostrará un diálogo en caso de que algún campo no se haya completado y se haya pulsado el botón de "Siguiente". |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | |

Tabla 92: RS-38

| IDENTIFICADOR: RS-39 | |
|---|---|
| **Descripción:** | El sistema de información mostrará un diálogo en el módulo de lectura cuando se necesite una contraseña para descifrar el mensaje. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | RUC-18, RUR-10. |

Tabla 93: RS-39

| IDENTIFICADOR: RS-40 | |
|---|---|
| **Descripción:** | El sistema de información mostrará un diálogo en el módulo de lectura cuando el mensaje contenga una URL dando a elegir al usuario entre que se le redireccione o no. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | RUC-20. |

Tabla 94: RS-40

### 4.5.3 REQUISITOS DE INTERFAZ

| IDENTIFICADOR: RS-41 | |
|---|---|
| **Descripción:** | La aplicación podrá ser llamada de manera externa cuando se seleccione desde el menú contextual de Android “abrir con” sobre una imagen que contenga un código QR o un código QRypt. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☑ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-23. |

Tabla 95: RS-41

| IDENTIFICADOR: RS-42 | |
|---|---|
| **Descripción:** | El sistema de información debe permitir al usuario poder seleccionar imágenes con formato JPEG y PNG. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-05, RUC-23. |

Tabla 96: RS-42

### 4.5.4 REQUISITOS DE RENDIMIENTO

| IDENTIFICADOR: RS-43 | |
|---|---|
| **Descripción:** | El sistema de información debe permitir al usuario poder seleccionar imágenes de cualquier resolución, peso y tamaño. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | RUC-05, RUC-23. |

Tabla 97: RS-43

| IDENTIFICADOR: RS-44 | |
|---|---|
| **Descripción:** | El sistema de información debe ser capaz de incluir un mensaje en un código QR en menos de 30 segundos y cifrar e incluir un mensaje cifrado en un código QRypt en menos de 1 minuto. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | RUR-07. |

Tabla 98: RS-44

| IDENTIFICADOR: RS-45 | |
|---|---|
| **Descripción:** | La aplicación se mostrará en castellano o inglés dependiendo del idioma del dispositivo Android. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☐ Primario ☐ Secundario ☑ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUR-06. |

Tabla 99: RS-45

### 4.5.5 REQUISITOS DE RECURSOS

| IDENTIFICADOR: RS-46 | |
|---|---|
| **Descripción:** | La aplicación debe disponer de conexión a Internet para cifrar y descifrar QRypt con la fecha como clave y para comprobar si existen nuevas actualizaciones que están disponibles. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | RUR-05. |

Tabla 100: RS-46

| IDENTIFICADOR: RS-47 | |
|---|---|
| **Descripción:** | La aplicación funcionará en sistemas operativos Android a partir de la versión de firmware 3.0. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | |

Tabla 101: RS-47

| IDENTIFICADOR: RS-48 | |
|---|---|
| **Descripción:** | El terminal deberá contar con la memoria RAM suficiente para poder utilizar el sistema sin ninguna excepción. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | |

Tabla 102: RS-48

| IDENTIFICADOR: RS-49 | |
|---|---|
| **Descripción:** | El terminal deberá contar con una cámara para poder escanear códigos QR desde ella. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | RUC-03. |

Tabla 103: RS-49

### 4.5.6 REQUISITOS DE SEGURIDAD

| IDENTIFICADOR: RS-50 | |
|---|---|
| **Descripción:** | La aplicación deberá utilizar algoritmos de cifrado seguros. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☐ Primario ☑ Secundario ☐ Opcional |
| **Estabilidad:** | ☐ Alta ☑ Baja |
| **Requisitos:** | RUR-04. |

Tabla 104: RS-50

| IDENTIFICADOR: RS-51 | |
|---|---|
| **Descripción:** | La aplicación deberá asegurar que el mensaje cifrado introducido dentro de un QRypt sólo es posible descifrarlo si se conoce la contraseña que se utilizó cuando se cifró. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUR-03. |

Tabla 105: RS-51

| IDENTIFICADOR: RS-52 | |
|---|---|
| **Descripción:** | La aplicación deberá cumplir la Ley Orgánica de Protección de Datos. |
| **Prioridad:** | ☑ Alta ☐ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | |

Tabla 106: RS-52

### 4.5.7 REQUISITOS DE VERIFICACIÓN

| IDENTIFICADOR: RS-53 | |
|---|---|
| **Descripción:** | La aplicación tiene que controlar los errores que se puedan producir durante su funcionamiento y proporcionar los mensajes adecuados. |
| **Prioridad:** | ☐ Alta ☑ Media ☐ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | |

Tabla 107: RS-53

| IDENTIFICADOR: RS-54 | |
|---|---|
| **Descripción:** | La aplicación tiene que comprobar que se ha escrito un mensaje antes de intentar introducirlo en un código. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | |

Tabla 108: RS-54

| IDENTIFICADOR: RS-55 | |
|---|---|
| **Descripción:** | La aplicación tiene que comprobar que se ha elegido por lo menos un tipo de cifrado cuando se ha decidido generar un QRypt. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-24. |

Tabla 109: RS-55

| IDENTIFICADOR: RS-56 | |
|---|---|
| **Descripción:** | La aplicación tiene que comprobar que se ha elegido sólo un color cuando se ha decidido generar un QRypt. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-13. |

Tabla 110: RS-56

| IDENTIFICADOR: RS-57 | |
|---|---|
| **Descripción:** | La aplicación tiene que comprobar que se ha escogido una imagen en el modo de lectura por fichero. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-04, RUC-23. |

Tabla 111: RS-57

| IDENTIFICADOR: RS-58 | |
|---|---|
| **Descripción:** | La aplicación tiene que comprobar que la imagen contiene un código QR o un código QRypt. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-04, RUC-23. |

Tabla 112: RS-58

| IDENTIFICADOR: RS-59 | |
|---|---|
| **Descripción:** | La aplicación tiene que comprobar si el mensaje introducido sobrepasa el número de caracteres válidos máximo. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUR-01. |

Tabla 113: RS-59

| IDENTIFICADOR: RS-60 | |
|---|---|
| **Descripción:** | La aplicación tiene que comprobar si se ha introducido una contraseña, una fecha o un patrón en el supuesto de que se haya elegido uno o más de estos tipos de cifrado. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUC-07, RUR-10. |

Tabla 114: RS-60

| IDENTIFICADOR: RS-61 | |
|---|---|
| **Descripción:** | La aplicación tiene que comprobar que la contraseña introducida para descifrar el mensaje es válida y que se corresponde con la original. |
| **Prioridad:** | ☐ Alta ☐ Media ☑ Baja |
| **Necesidad:** | ☑ Primario ☐ Secundario ☐ Opcional |
| **Estabilidad:** | ☑ Alta ☐ Baja |
| **Requisitos:** | RUR-03. |

Tabla 115: RS-61

## 4.6 ANÁLISIS DE CLASES

En esta sección se van a obtener las clases que se necesitarán para el desarrollo de la aplicación a partir de los requisitos y casos de uso obtenidos anteriormente.

### 4.6.1 IDENTIFICACIÓN DE LAS CLASES

Como la aplicación debe desarrollarse para que funcione en el sistema operativo Android, hay que tener en cuenta que se necesita una clase por cada conjunto de interfaces de la aplicación; ya que, en Android, cada pantalla de la aplicación se diseña sobre un objeto Actividad, que no es más que un tipo de clase.

Las clases que se han obtenido de los requisitos y casos de uso han sido:

- **PantallaPrincipal:** Esta actividad contendrá la pantalla con los accesos principales (generar QR, generar QRypt, leer de cámara, leer de fichero). Además, también contendrá los métodos necesarios para modificar los ajustes y mostrar las novedades de la versión.
- **GeneradorQR:** Esta actividad contendrá las interfaces y los métodos que se utilizarán para obtener los datos que se usarán en la generación de los códigos QR y QRypt.
- **LectorQR:** Esta actividad contendrá la interfaz de muestra de mensaje leído (una interfaz donde se mostrará el mensaje que contenía el código QR o QRypt que se ha intentado leer) y los métodos necesarios para ello.
- **Camara:** Esta actividad contendrá la interfaz que se usa para poder enfocar un código mediante la cámara, y los métodos necesarios para extraer el mensaje y enviarlo a la clase *LectorQR*.
- **Utils:** Esta clase será la única de todas que no será una actividad, sino una clase de Java que importarán las actividades *GeneradorQR* y *LectorQR*. Esta clase contendrá los métodos necesarios para cifrar y descifrar mensajes, y para generar un código QR y leerlo de una imagen.

### 4.6.2 ESPECIFICACIÓN DE LAS FUNCIONES DE CADA CLASE

Una vez se han identificado las clases, se explicarán los métodos importantes de cada una:

- **PantallaPrincipal:**
  - **abrirGenerarQR:** Se encargará de ejecutar la actividad *GeneradorQR* con un valor en una variable que especificará que se quiere crear un código QR.
  - **abrirGenerarQRypt:** Se encargará de ejecutar la actividad *GeneradorQR* con un valor en una variable que especificará que se quiere crear un código QRypt.
  - **abrirLeerDeCamara:** Se encargará de ejecutar la actividad *Camara*.
  - **abrirLeerDeFichero:** Se encargará de ejecutar la actividad *LectorQR* con un valor en una variable que especificará que se debe abrir el explorador de archivos.
  - **modificarAjustes:** Se encargará de modificar los valores por defecto de los ajustes de la aplicación. Se accede desde el menú deslizable.

    - **novedadesVersion:** Se encargará de mostrar las novedades de la versión actual. Se accede desde el menú deslizable.

Como se puede observar, a la actividad *GeneradorQR* se le transferirá una variable para que sepa qué tipo de código se va a generar. En el caso de que se vaya a generar un código QRypt, la actividad tendrá que recoger más datos (contraseña y color) que en el caso de que se vaya a generar un QR.

- **GeneradorQR:**
    - **obtenerDatos:** Se encargará de pedirle al usuario los datos que se necesiten. Si se va a generar un QR, se pedirá el mensaje. Si se va a generar un QRypt, se pedirá el mensaje, la contraseña y el color. Estos datos se enviarán a la clase *Utils* para que cifre el mensaje y genere el código.
    - **mostrarCodigo:** Se encargará de mostrar por pantalla el código generado (ya sea QR o QRypt) para que el usuario pueda verlo.
    - **compartir:** Se encargará de enviar el código a la aplicación que se especifique.
    - **modificarAjustes:** Se encargará de modificar los valores por defecto de los ajustes de la aplicación. Se accede desde el menú deslizable.
    - **novedadesVersion:** Se encargará de mostrar las novedades de la versión actual. Se accede desde el menú deslizable.
- **Camara:**
    - **resetearCamara:** Se encargará de reiniciar la actividad si la cámara se congela por algún casual.
    - **identificarQR:** Se encargará de buscar en la imagen que se esté obteniendo en ese momento un código QR o QRypt.
    - **extraerMensaje:** Se encargará de extraer el mensaje del código QR que se haya identificado y enviarlo a la actividad *LectorQR*.
- **LectorQR:**
    - **mostrarMensaje:** Se encargará de imprimir el mensaje obtenido del código QR.
    - **obtenerDatos:** Se encargará de pedirle al usuario los datos que se necesiten. Si el mensaje está cifrado, se pedirá la contraseña. Estos datos se enviarán a la clase *Utils* para que descifre el mensaje.
    - **modificarAjustes:** Se encargará de modificar los valores por defecto de los ajustes de la aplicación. Se accede desde el menú deslizable.
    - **novedadesVersion:** Se encargará de mostrar las novedades de la versión actual. Se accede desde el menú deslizable.
- **Utils:**
    - **dibujarQR:** Se encargará de generar un código QR a partir de un texto (cifrado o no) y un color.
    - **leerArchivo:** Se encargará de abrir el explorador de archivos y extraer el mensaje del código QR o QRypt que se encuentre representado en la imagen que el usuario haya escogido. *LectorQR* lo ejecutará cuando necesite el mensaje de una imagen.

- **cifrar:** Se encargará de cifrar un mensaje con una contraseña.
- **descifrar:** Se encargará de descifrar un mensaje con una contraseña.

Queda aclarar que, cuando se elija en la pantalla principal que se quiere leer un archivo de imagen, ejecuta la actividad *LectorQR* pero con una variable que avisa de que se tiene que ejecutar el explorador de archivos para recoger la imagen. Cuando se ejecute *LectorQR*, esta actividad ejecutará la función *leerArchivo* de la clase *Utils*.

### 4.6.3 DIAGRAMA DE CLASES

Una vez se han identificado las clases, se mostrará, a continuación, cómo se relacionan entre ellas:

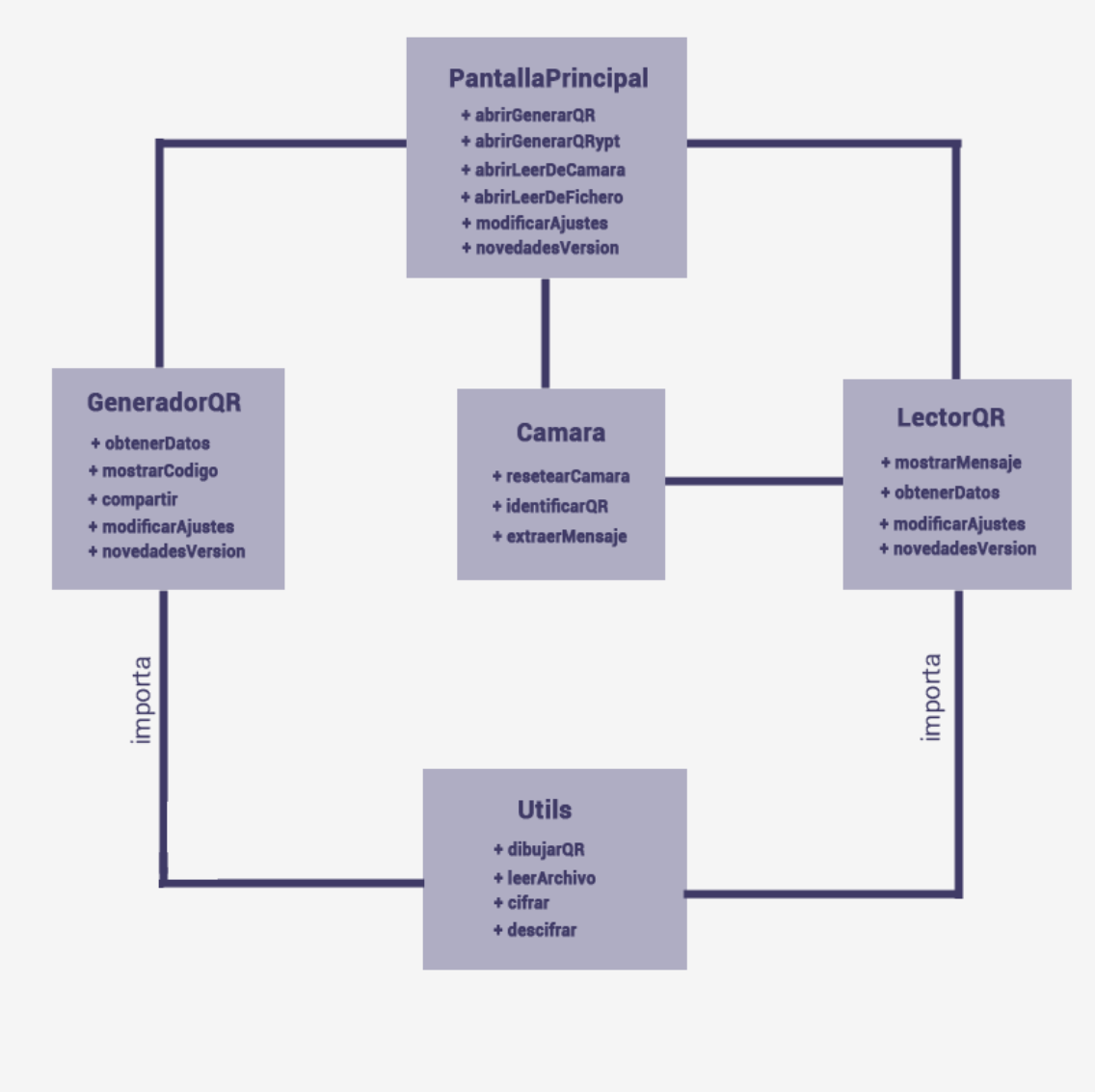



Ilustración 8: Diagrama de clases

El diagrama representado en la ilustración anterior es muy simple. Ninguna de las clases que se van a implementar heredará de ninguna otra. Lo único que harán será apoyarse unas en otras para obtener los resultados que el usuario desea.

De esta forma, se puede observar que la clase *PantallaPrincipal* podrá ejecutar las clases *GeneradorQR*, *Camara* y *LectorQR*. La clase *GeneradorQR* podrá volver a la clase *PantallaPrincipal* y podrá inicializar un objeto *Utils* (importando la clase) ya que ésta posee unos métodos que le harán falta. La clase *Camara* podrá ejecutar la clase *LectorQR* (cuando obtenga el mensaje del código QR enfocado) y, también, podrá volver a la clase *PantallaPrincipal*. Por último, la clase *LectorQR* podrá volver a la clase *PantallaPrincipal* y podrá inicializar un objeto *Utils* (importando la clase) de la misma forma que lo hará la clase *GeneradorQR*.

# 5 DISEÑO

Esta sección pretende resolver el problema que se ha analizado y descrito en el Análisis. Además, se justificarán todas y cada una de las decisiones que se tomen.

Como se ha hecho anteriormente y, para facilitar su comprensión, se va a seguir un procedimiento incremental para realizar el diseño del sistema. En este procedimiento se empezará describiendo y explicando la aplicación y, más tarde, se irán descomponiendo los subsistemas que la forman hasta llegar a cada uno de los detalles específicos de cada módulo.

## 5.1 ARQUITECTURA DEL SISTEMA

Para realizar este apartado se ha decidido dividir el sistema en capas que se comunican con la que tienen debajo. En la Ilustración 9 se puede observar la arquitectura.

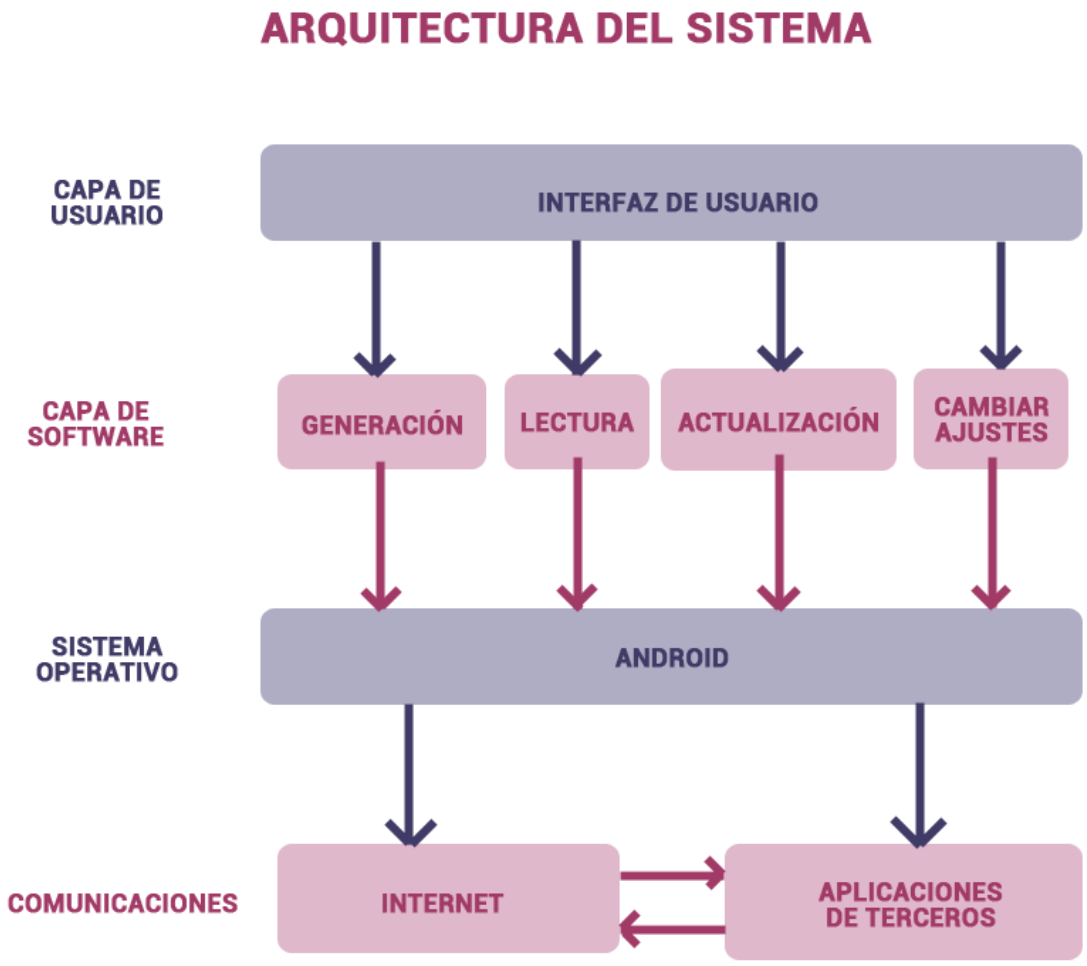



Ilustración 9: Diagrama de la arquitectura del sistema

Como se puede ver en la Ilustración 9, la arquitectura del sistema cuenta con 4 capas: la capa usuario, la capa de software, el sistema operativo y las comunicaciones. Todas estas capas se conectan con la que está inmediatamente debajo si es que existe.

La arquitectura del sistema comienza en la capa de usuario que abarca toda la interfaz del sistema. Esta capa se comunicará con los 4 módulos que son los que contienen las funcionalidades de la aplicación. Además, el sistema de información llamará al sistema operativo Android para poder lograr su objetivo. Finalmente, se comunicará con las aplicaciones de terceros para poder compartir el código que se ha generado y con Internet para comprobar las actualizaciones de la aplicación. Por último, es preciso recalcar que muchas de las aplicaciones de terceros también necesitan una conexión a Internet para su correcto funcionamiento.

Seguidamente, se procederá a realizar una descripción de cada una de las capas que componen la arquitectura del sistema que se tiene que diseñar:

- **Interfaz de usuario:** Su función consiste en otorgar al usuario una comunicación con el sistema. A partir de esta capa, se podrá acceder a todos los módulos de la capa inmediatamente más baja: la capa de software. Además, los datos que precise la aplicación para lograr su objetivo serán introducidos por el usuario gracias a esta capa. Debido a su importancia, esta capa debe estar bien diseñada de forma que sea accesible y usable para el usuario.
- **Generación de códigos QR y QRypt:** Siempre y cuando se parta de unos datos suministrados por el usuario a través de la capa superior (la interfaz de usuario), este módulo debería ser capaz de incluir un mensaje, cifrado o no, en un código QR o QRypt para, posteriormente, compartirla con otras aplicaciones instaladas en el dispositivo.
- **Lectura de códigos QR y QRypt:** Este módulo estará encargado de extraer un mensaje de un código QR o QRypt. Si este mensaje estuviese cifrado, debería poder descifrarlo con la contraseña que el usuario le otorgue.
- **Actualización:** Este módulo debería permitir al usuario que se informe de las novedades que incluye la versión de la aplicación que está instalada en su dispositivo.
- **Cambiar ajustes:** Este módulo debería permitir al usuario que cambie los ajustes por defecto de la aplicación, como la carpeta de salida o la contraseña por defecto.
- **Sistema operativo Android:** Los módulos anteriores deberán comunicarse con el sistema operativo del terminal. Esta capa será la encargada de ejecutar todas las llamadas al sistema que el sistema realice.
- **Comunicación externa:** Todos los códigos que sean generados por medio de la aplicación podrán ser compartidos con otras aplicaciones instaladas en el terminal. Para que esto sea posible, tanto las aplicaciones de terceros como el sistema que se va a desarrollar deben establecer vías de comunicación entre ellos por medio de los mecanismos nativos que posee el sistema operativo

Android. Además, será necesaria una conexión a Internet para que el módulo de actualización tenga efecto.

Como se puede deducir, los módulos principales del sistema son los que tengan que ver con la generación de códigos y con la lectura de ellos. Estos módulos serán los primeros en diseñarse e implementarse. Los módulos de actualización y cambio de ajustes quedarán como módulos opcionales y se realizarán después, si los anteriores módulos se resuelven sin problema y aun queda tiempo para implementar estas funcionalidades.

## 5.2 SUBSISTEMAS

Gracias a la arquitectura que se ha propuesto para el sistema en el apartado anterior y a los requisitos que se han obtenido y definido en el Análisis, se ha decidido dividir el sistema en subsistemas para facilitar el proceso de diseño del mismo.

Para el sistema que se va a diseñar, los subsistemas coinciden con los módulos de la capa de software que se han especificado en la arquitectura del sistema; ya que, comúnmente, un subsistema se reconoce por los servicios que ofrece. Además, estos servicios forman un compuesto de funciones con una intención común.

Los subsistemas en los que se ha decidido fraccionar el sistema y los cuáles se procederán a diseñar son: generación, lectura, actualización y cambio de ajustes.

## 5.3 SUBSISTEMA DE GENERACIÓN

El objetivo del módulo de generación es generar un código que cumpla el estándar de QR que porte un mensaje a partir de las especificaciones del usuario. El mensaje puede estar cifrado con diferentes claves de forma que su mensaje quedará protegido y sólo podrán acceder a él las personas autorizadas. Como se pueden generar dos tipos diferentes de código y su proceso de generación dista bastante uno del otro, se va a diseñar cada uno por separado.

### 5.3.1 GENERACIÓN DE CÓDIGOS QR

Lo primero que se debe realizar es una especificación de la estructura del módulo de generación de códigos QR para conocer las funciones importantes que se deben diseñar.


Ilustración 10: Diagrama de la estructura del módulo de generación QR

En la Ilustración 10 se observan los 3 componentes principales que actúan en el proceso de generación de un código QR con un mensaje del usuario. A continuación, se procederá a describir por separado cada uno de éstos.

#### *5.3.1.1 RECOGER PARÁMETROS*

La aplicación permite al usuario comunicarse con su terminal con el fin de especificar los parámetros que el sistema necesita para la generación de estos tipos de código.

Como se trata únicamente de un código de naturaleza QR, el único parámetro que la aplicación necesita recoger del usuario es el mensaje que éste quiere incluir en el código.

Una vez se ha introducido el mensaje, la aplicación lo validará comprobando:

- Si el mensaje introducido no contiene caracteres no permitidos.
- Si se ha introducido un mensaje (la cadena no está vacía).

Si se da error en una de estas validaciones, el sistema se lo hará saber al usuario por medio de diálogos emergentes.

En la siguiente tabla, se van a exponer los atributos que obtendrá el sistema junto a sus tipos representados en lenguaje de programación Java y los valores que éstos pueden tomar a nivel interno:

| Parámetro | Tipo | Valor |
|---|---|---|
| **Mensaje** | String | 1-2000 caracteres |

Tabla 116: Parámetros del módulo de generación QR

#### 5.3.1.2 GENERACIÓN DEL CÓDIGO QR

Una vez se obtienen los parámetros, la aplicación pasará a generar el código QR. Para ello, el sistema deberá transformar el mensaje introducido por el usuario a una matriz de bits compatible con la ayuda de una librería de generación de códigos QR para después transformar esa matriz en un objeto Bitmap, que es el objeto imagen de Android. Una vez se tiene el Bitmap, ya es posible transformarlo en un archivo PNG y, así, poder escanear más tarde el código. En la Ilustración 11 se puede observar un diagrama en el que figuran de forma básica lo que se acaba de explicar.

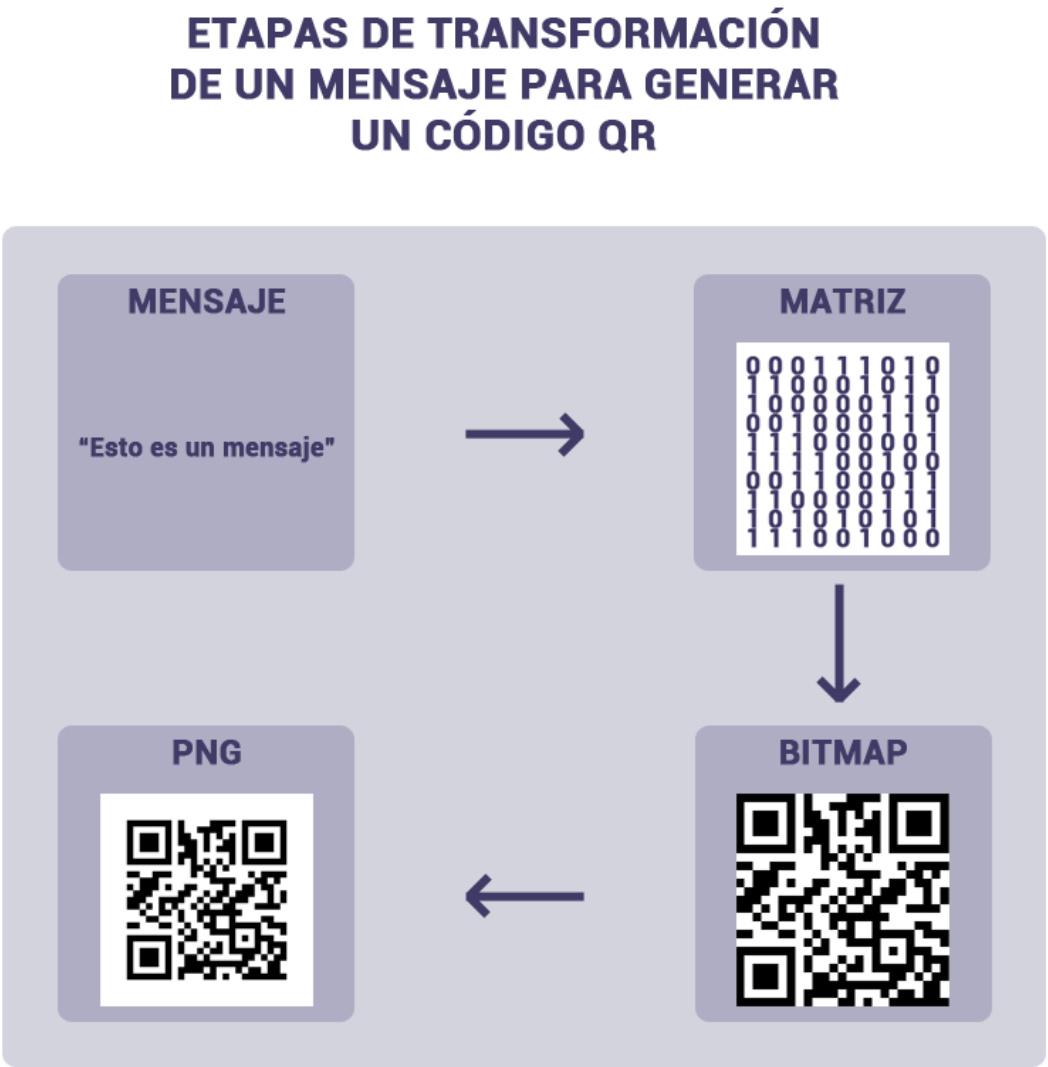



Ilustración 11: Etapas de transformación de un mensaje para generar un código QR

A continuación, se explicará más en detalle cómo se trata el elemento de una etapa para dar como resultado el de la siguiente:

1. Una vez se obtiene el mensaje, se transformará la cadena por medio de una librería externa llamada ZXing en una matriz de bits. ZXing es la librería estándar de Java para tratar códigos de barras. A pesar de no ser nativa de Java, sí que es la más utilizada por su rapidez y eficiencia. Por estas razones, se ha considerado oportuno elegirla para utilizarla en el presente proyecto.
2. Cuando se obtenga la matriz de bits del mensaje será necesario transformarla a un objeto Bitmap. Para ello, se irá leyendo cada posición de la matriz y se irá escribiendo cada pixel del Bitmap en blanco o en negro dependiendo del dato que se haya leído.
3. Por último, una vez se obtenga el Bitmap con el mensaje introducido en un código QR, se generará un archivo PNG con las clases y funciones de Java a partir de éste. Esta imagen se guardará en una carpeta de la tarjeta SD del dispositivo con un nombre de archivo basado en la fecha y hora a la que se generó el código, de esta forma no habrá nunca dos archivos que se llamen igual.

La implementación de estas etapas será definida y explicada en el capítulo de implementación del sistema. En ese capítulo se entrará en detalles y se especificarán los algoritmos que se emplearán.

#### *5.3.1.3 ENVÍO*

La última tarea del módulo enviará la imagen PNG a una aplicación de terceros instalada en el dispositivo para que ésta la trate. Las aplicaciones de terceros a las que se podrá mandar la imagen serán aquellas que aparezcan en el menú de Android "Compartir con", que son las que están instaladas en el dispositivo y permiten que se les comparta imágenes para procesarlas ellas.

Algunos ejemplos de estas aplicaciones que la gran mayoría de usuarios Android tiene son:

- **Email:** Tanto GMail como EMail, permiten al usuario sincronizar una o más cuentas de correo electrónico para recibir los mensajes nuevos en el dispositivo y, también, para crearlos. Estas dos aplicaciones permiten recoger una imagen para adjuntarla a un nuevo correo; por lo tanto, cuando se seleccione esta opción en el menú compartir de la aplicación que se va a desarrollar, se abrirá un nuevo correo y se adjuntará el código a él. De esta forma, no es necesario adjuntarla desde la galería.
- **Dropbox:** La aplicación de Dropbox es la versión móvil de la famosa aplicación web de almacenamiento en la nube. Esta aplicación permite subir archivos a una cuenta de Dropbox, descargarlos y moverlos de carpeta. Si se selecciona la aplicación Dropbox desde el menú compartir de la aplicación que se va a desarrollar, se podrá subir la imagen a la cuenta Dropbox y se podrá decidir la carpeta donde se hará.
- **Google Drive:** Es la aplicación oficial del servicio web de Google de almacenamiento en la nube. Esta aplicación permite subir archivos a una cuenta de Google, descargarlos y moverlos de carpeta. Si se selecciona la aplicación Google Drive desde el menú compartir de la aplicación que se va a

desarrollar, se podrá subir la imagen a la cuenta Google y se podrá decidir la carpeta donde se hará.

- **Whatsapp:** Es la aplicación de mensajería más utilizada en el mundo. Además de mensajes, permite enviar imágenes, vídeos y archivos de audio. Si se selecciona la aplicación Whatsapp desde el menú compartir de la aplicación que se va a desarrollar, se podrá compartir la imagen con cualquiera de los contactos que existan en la cuenta Whatsapp que esté dada de alta en el terminal.
- **Line:** Esta aplicación es idéntica a Whatsapp salvo que, además, permite realizar llamadas por VoIP. Al igual que en el Whatsapp si se selecciona la aplicación Line desde el menú compartir de la aplicación que se va a desarrollar, se podrá compartir la imagen con cualquiera de los contactos que existan en la cuenta Line que esté dada de alta en el terminal.
- **Facebook:** Esta aplicación es la aplicación oficial del servicio web homónimo de naturaleza social. Con esta aplicación se puede cambiar el estado del perfil de la cuenta que esté autorizada en el terminal, subir fotos, comentar en otros perfiles y casi todas las opciones que ofrece el servicio web. Si se selecciona la aplicación Facebook desde el menú compartir de la aplicación que se va a desarrollar, se podrá subir o publicar la imagen en el muro de la cuenta de Facebook que esté registrada en el terminal.
- **Twitter:** Al igual que Facebook, Twitter es un servicio web que ofrece una gran cantidad de funciones de naturaleza social. Con esta aplicación se pueden publicar tweets, contestar los tweets de otras personas, empezar a seguir a otras cuentas y, generalmente, lo mismo que ofrece el servicio web. Si se selecciona la aplicación Twitter desde el menú compartir de la aplicación que se va a desarrollar, se podrá publicar un tweet con el código adjunto a él.

Además de las aplicaciones que se han descrito, existen muchas más que pueden tratar la imagen (todas las que estén programadas para recibir por parámetros una archivo de imagen). También funcionaría con servicios de Android como el Bluetooth y los MMS, los cuales, eligiéndolos en el menú compartir de la aplicación que se va a desarrollar, lo transferirían a un usuario y lo adjuntarían a un mensaje, respectivamente.

### 5.3.2 GENERACIÓN DE CÓDIGOS QRYPT

Lo primero que se debe realizar es una especificación de la estructura del módulo de generación de códigos QR para conocer las funciones importantes que se deben diseñar.

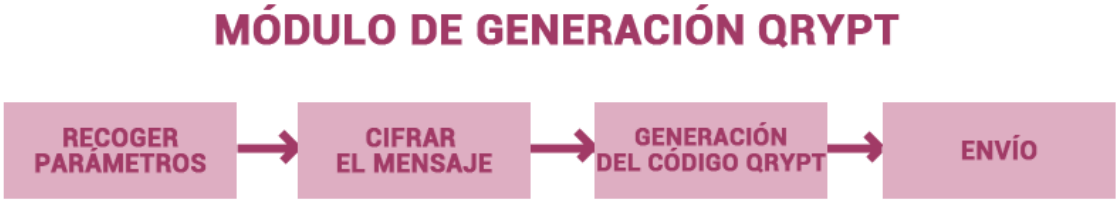



Ilustración 12: Diagrama de la estructura del módulo de generación QRypt

En la Ilustración 12 se observan los 4 componentes principales que actúan en el proceso de generación de un código QRypt con un mensaje del usuario. A continuación, se procederá a describir por separado cada uno de éstos.

#### *5.3.2.1 RECOGER PARÁMETROS*

La aplicación permite al usuario comunicarse con su terminal con el fin de especificar los parámetros que el sistema necesita para la generación de estos tipos de código.

Como se trata de un código de naturaleza QRypt, la aplicación necesita recoger más parámetros además del mensaje que el usuario quiere incluir en el código. Los atributos que el sistema necesita del usuario para generar un código QRypt son:

- Un mensaje que será incluido en el código QRypt.
- Un tipo de cifrado para proteger el mensaje. Los tipos de cifrado que el programa debe ofrecer al usuario son los siguientes:
  - Contraseña (Se necesitará especificar cuál).
  - Imei del dispositivo.
  - Modelo del terminal.
  - Fecha (Se necesitará especificar cuál).
  - Patrón (Se necesitará especificar cuál).
- Un color para que el código QRypt se diferencia de un código QR normal.

Una vez se han introducido los parámetros, la aplicación los validará comprobando:

- Si el mensaje introducido ni la contraseña (en caso de que se haya consultado) no contiene caracteres no permitidos.
- Si se ha introducido un mensaje (la cadena no está vacía).
- Si, en caso de que se haya elegido el tipo de cifrado por contraseña, se ha introducido una.
- Si, en caso de que se haya elegido el tipo de cifrado por fecha, se ha introducido una.
- Si, en caso de que se haya elegido el tipo de cifrado por patrón, se ha introducido uno.
- Si se ha escogido un color.

Si se da error en una de estas validaciones, el sistema se lo hará saber al usuario por medio de diálogos emergentes.

En la siguiente tabla, se van a exponer los atributos que obtendrá el sistema junto a sus tipos representados en lenguaje de programación Java y los valores que éstos pueden tomar a nivel interno:

| Parámetro | Tipo | Valor |
|---|---|---|
| **Mensaje** | String | 1-2000 caracteres |
| **Contraseña** | String | Más de un caracter |
| **Color** | Integer | 0: Verde, 1: Rojo, 2: Azul, 3: Naranja, 4: Morado, 5: Marrón, 6: Rosa, 7: Gris, 8: Random |

Tabla 117: Parámetros del módulo de generación QRypt

#### 5.3.2.2 CIFRAR EL MENSAJE

Este proceso se encargará de implementar los algoritmos criptográficos necesarios para cifrar el mensaje que se va a introducir en el código QRypt. Esta tarea se divide en 4 subtareas que cifrarán el mensaje de tal forma que sea casi imposible obtener el mensaje original si no se posee la información necesaria para descifrarlo. Las subtareas de las que se compone son las siguientes:


Ilustración 13: Diagrama general de las etapas del cifrado de un mensaje

- **Cifrado:** El mensaje se cifrará utilizando el algoritmo AES con una clave de 256 bits que derivará de la contraseña. Esta clave se obtendrá derivando con la función PBKDF2 (Password-Based Key Derivation Function 2). Esta última función derivará más de 1000 veces la contraseña, que no será otra que los hashes de seguridad SHA-1 de la opciones de seguridad concatenados, utilizando algoritmos criptográficos y funciones resumen. Esta derivación no tiene otra función que la de aumentar el tiempo de cómputo y, así, perjudicar un ataque de fuerza bruta. Al mensaje cifrado se le concatenará uno o más caracteres especiales dependiendo de las opciones de cifrado seleccionadas. De esta forma, en el proceso de descifrado se podrá saber cómo se cifró el mensaje.

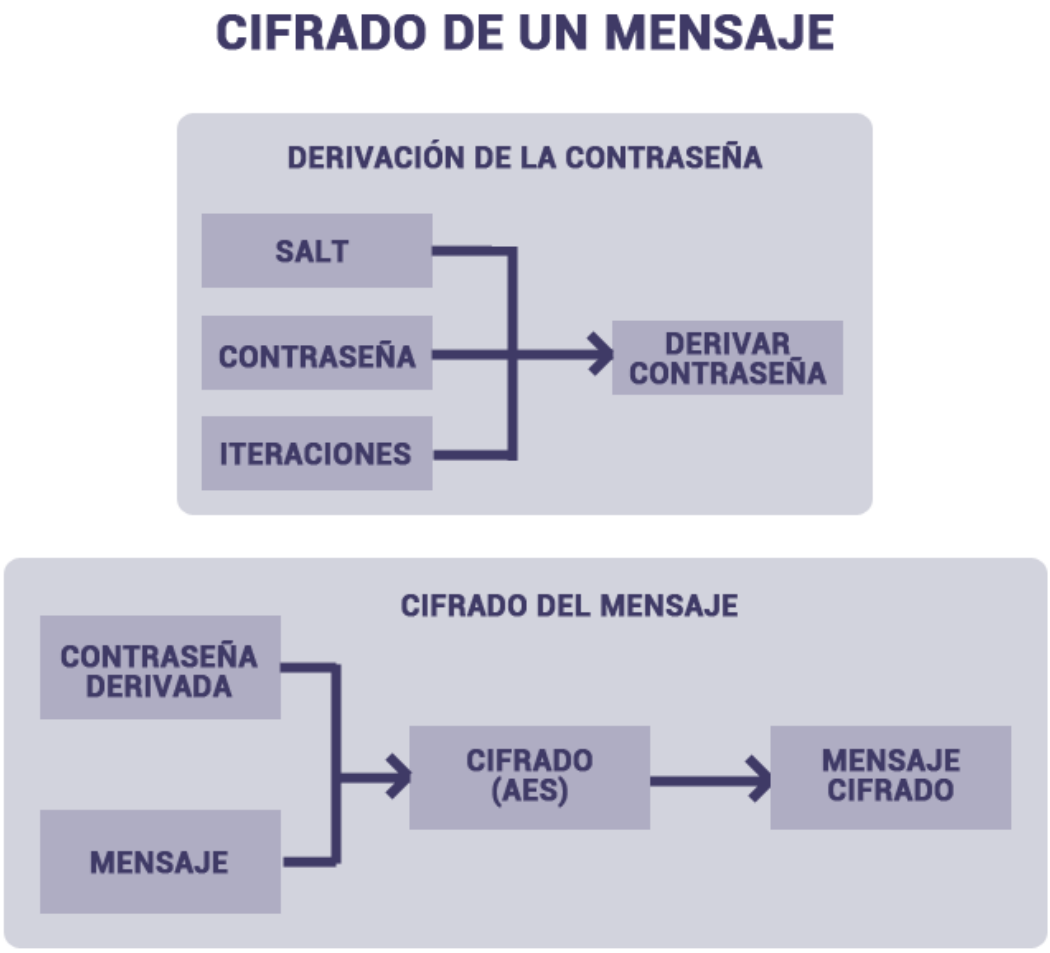



Ilustración 14: Diagrama general de la derivación de la contraseña y del cifrado del mensaje

A la hora de derivar la contraseña, primero se concatenará un *salt* aleatorio con la contraseña y esto se derivará con PBKDF2 un número aleatorio de iteraciones.

Por otro lado, cuando se cifre el mensaje se utilizará la contraseña que se ha derivado y se cifrará el mensaje con el algoritmo AES utilizando CBC como tipo de cifrado para el bloque y se generará un vector de inicialización con tamaño de bloque AES. De esta forma, siempre se obtendrá un mensaje distinto aunque se utilice la misma contraseña.

Por último, se concatenará al mensaje cifrado el *salt*, el vector de inicialización y el número de iteraciones para poder descifrarlo de manera óptima.

- **Desordenamiento:** Además de cifrarlo, la aplicación desordenará el mensaje con el único fin de provocar confusión en el atacante. El algoritmo que se ejecuta en este proceso sustituirá las posiciones de los caracteres del mensaje de forma que, cuando se vaya a descifrar, sea reversible.
- **Codificación:** Una vez se cifre el texto y desordene, se codificará en *base64*. Se ha elegido esta opción ya que, de esta forma, se permitiría representar el texto

resultante en caracteres *ASCII* y se eludirían problemas en el proceso de descifrado y generación del código.

- **Tokens:** Una vez se ha cifrado, desordenado y codificado el mensaje, la última subtarea que se llevará a cabo será la de concatenar al mensaje unos tokens de control para conocer los tipos de cifrado que se han empleado y, así, poder generar la contraseña en el proceso de descifrado. La siguiente tabla muestra los caracteres que se concatenarán y el tipo de cifrado al que representan.

| TOKENS DE CONTROL | |
|---|---|
| % | Contraseña |
| $ | Imei |
| # | Modelo |
| @ | Fecha |
| & | Patrón |

Tabla 118: Tokens de control

#### 5.3.2.3 GENERACIÓN DEL CÓDIGO QRYPT

Esta etapa se desarrolla de la misma forma que en los códigos QR, salvo que, en vez de obtener la matriz de bits del mensaje, se obtiene del mensaje cifrado. Y, en lugar de utilizar los colores blanco y negro, se utilizará el color elegido por el usuario y blanco. Por eso, no se considera necesario explicar más en detalle cómo se va a realizar esta función ya que sería repetir otra vez el mismo apartado.

#### 5.3.2.4 ENVÍO

Al igual que el apartado anterior, esta función es completamente idéntica a la que se ha definido en la sección homónima de generación de códigos QR. Por esta razón, no se va a ahondar más en el tema.

## 5.4 SUBSISTEMA DE LECTURA

El objetivo del módulo de lectura es extraer un mensaje de un código QR o QRypt. El mensaje puede estar cifrado con diferentes claves de forma que quedará protegido y sólo podrán acceder a él las personas autorizadas. Como se pueden leer los códigos de dos formas que distan mucho una de otra, se va a diseñar cada una por separado.

### 5.4.1 LECTURA POR CÁMARA

Lo primero que se debe realizar es una especificación de la estructura del módulo de lectura de códigos QR o QRypt por cámara para conocer las funciones importantes que se deben diseñar.

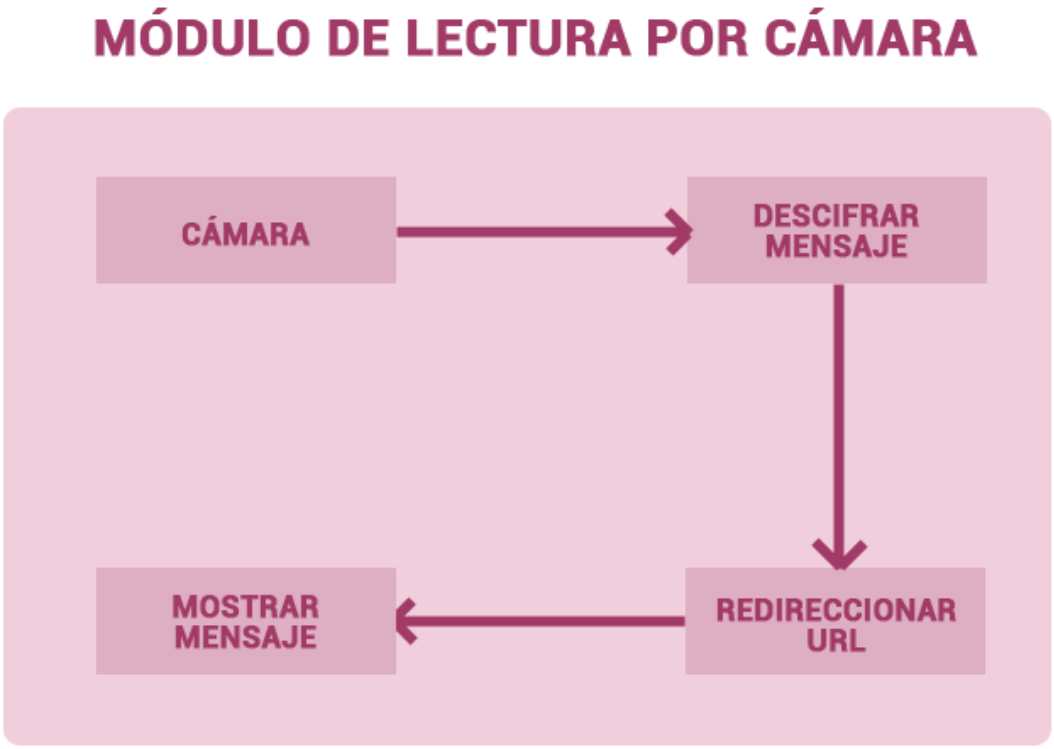



Ilustración 15: Diagrama de la estructura del módulo de lectura por cámara

En la Ilustración 15 se observan los 4 componentes principales que actúan en el proceso de lectura de un código QR o QRypt por cámara. A continuación, se procederá a describir por separado cada uno de éstos.

#### 5.4.1.1 CÁMARA

La aplicación permite al usuario hacer uso de la cámara del dispositivo móvil donde está instalada con el fin de enfocar un código de naturaleza QR o QRypt y extraer el mensaje que éste tiene almacenado en su interior.

El funcionamiento de esta etapa vendrá soportado por una librería de descifrado de códigos QR en tiempo real para una ejecución más rápida. Esta librería extraerá el mensaje del código QR que está enfocando en ese momento la cámara. Una vez se ha extraído el mensaje, se podrá comprobar si está cifrado o no buscando en él alguno

de los caracteres especiales que lo identifiquen como tal. La librería que se utilizará no será la ZXing porque, a pesar de que tiene esta funcionalidad, obliga al usuario a tener instalada su aplicación principal en el terminal ya que, para hacer uso de la cámara, se tiene que hacer una llamada a su aplicación y ésta devuelve la información del QR.

Además, si por algún casual, la cámara dejase de responder, se pondrá a disposición del usuario un botón que la reiniciará para poder seguir enfocando el código QR sin que la aplicación fuerce un cerrado. Si se da algún error interno que no sea éste y no se pueda reiniciar el proceso de la cámara, la aplicación devolverá al usuario a la pantalla principal y mostrará un mensaje de error.

#### *5.4.1.2 DESCIFRAR EL MENSAJE*

Este proceso sólo se ejecutará si el mensaje obtenido en el proceso anterior contiene alguno de los caracteres especiales; es decir, está cifrado.

Se ha decidido descomponer esta etapa en subprocesos para facilitar su comprensión. En la siguiente Ilustración se expone un diagrama que muestra estos subprocesos y cómo se relacionan entre ellos para que, cuando se explique cada uno, se tenga como referencia.


Ilustración 16: Diagrama general de las etapas del descifrado de un mensaje

A continuación se procederá a explicar cada una de las etapas (o subprocesos) que figuran en el diagrama:

- **Decodificación:** Será igual que el proceso de codificación en *base64* que se utilizó anteriormente en el proceso de generación de un código QRypt pero invertido. En esta etapa se obtendrán los caracteres del mensaje no codificados. Aun faltará ordenarlos y descifrarlos.
- **Ordenación:** Se utilizará el mismo algoritmo de desordenamiento que en el proceso de generación de un código QRypt pero, al igual que en la etapa anterior, invertido.
- **Descifrado:** En esta etapa se descifrará el mensaje que ya estará decodificado y ordenado. Se va a utilizar el mismo procedimiento que en el proceso de cifrado, empleando el mismo algoritmo AES y la misma derivación de contraseña. Se observarán los caracteres especiales para saber con qué método se ha cifrado y, en el caso de haber elegido por patrón o por contraseña, se pedirá al usuario la contraseña. En los demás casos, al ser datos que ya figuran en el dispositivo, se podría realizar el subproceso de

descifrado sin ningún tipo de comunicación con el usuario. Como en el proceso de cifrado se concatenaría al mensaje el *salt*, el vector de inicialización y el número de iteraciones para la derivación de la contraseña, todo este proceso se realizaría de la misma manera que en el cifrado.


Ilustración 17: Diagrama general de la derivación de la contraseña y del descifrado del mensaje

Después de ejecutar estos subprocesos, el mensaje quedará completamente descifrado y listo para ser mostrado pero, antes, se realizará una última comprobación.

#### *5.4.1.3 REDIRECCIONAR URL*

Si el mensaje descifrado cumple los requisitos para ser una URL, se preguntará al usuario si quiere ser redireccionado a esa página web. En el mensaje que se le mostrará al usuario estará incluida la URL para que el usuario pueda saber con antelación a dónde se le va a redirigir si acepta.

Los requisitos mínimos que se tienen que cumplir para que el mensaje sea considerado una URL son los siguientes:

- El mensaje empieza por "http://", "https://" o "ftp://".

- Si no se cumple el caso de arriba, el mensaje empieza por "www." o "ftp.".
- En alguna posición de mensaje existe un punto (.) seguido de un texto (es) y una barra (/). La barra puede ser, también, directamente el final del mensaje. De esta forma, se detectan las URLs que son de este tipo "url.es", "url.es/" y "url.es/pagina1".

Aunque el usuario elija que se le redirija al enlace, la aplicación realizará el siguiente proceso; en un caso, abrirá el navegador encima y no se verá el mensaje hasta que no se cierre, y en el otro, se mostrará directamente.

#### *5.4.1.4 MOSTRAR MENSAJE*

Una vez se tenga el mensaje y sin importar si es una URL o no, la aplicación mostrará su contenido para satisfacer la necesidad del usuario.

### 5.4.2 LECTURA POR FICHERO

Lo primero que se debe realizar es una especificación de la estructura del módulo de lectura de códigos QR o QRypt por fichero para conocer las funciones importantes que se deben diseñar.

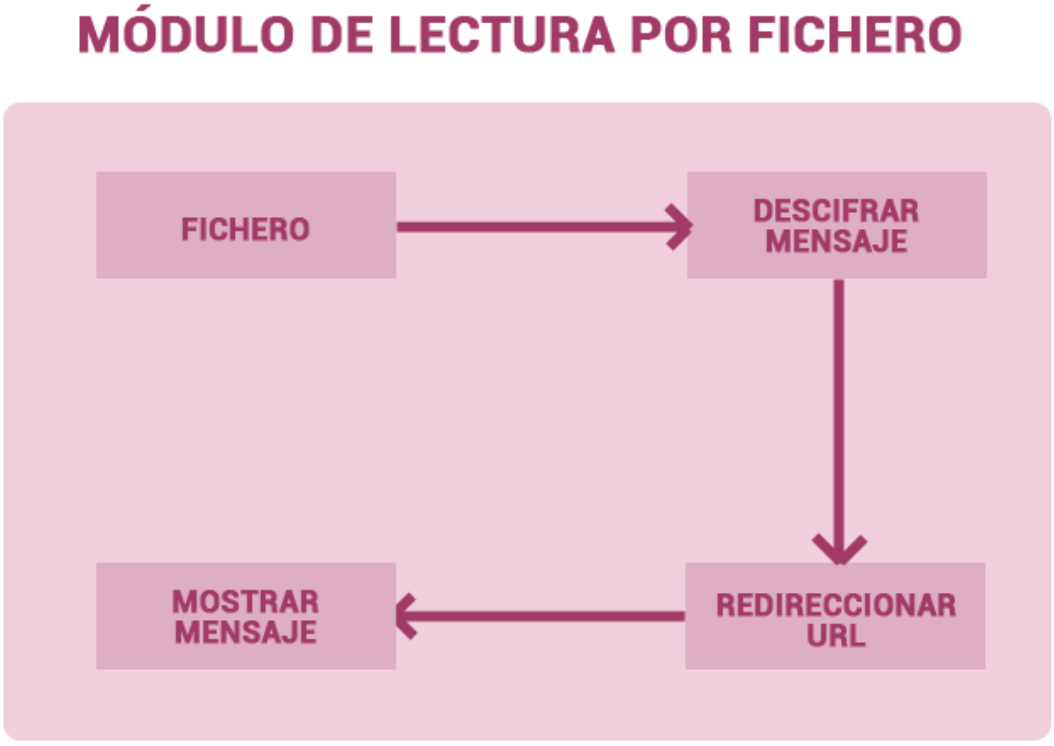



Ilustración 18: Diagrama de la estructura del módulo de lectura por fichero

En la Ilustración 18 se observan los 4 componentes principales que actúan en el proceso de lectura de un código QR o QRypt por cámara. A continuación, se procederá a describir por separado cada uno de éstos.

#### 5.4.2.1 FICHERO

La aplicación permite al usuario hacer uso del explorador de archivos del dispositivo móvil donde está instalada con el fin de seleccionar un archivo que contenga un código de naturaleza QR o QRypt y extraer el mensaje que éste tiene almacenado en su interior.

El funcionamiento de esta etapa vendrá soportado por una librería de descifrado de códigos QR cuando éstos se encuentran en un archivo de imagen. Esta librería reconocerá por patrones dónde se encuentra el código QR y lo aislará de resto de la imagen para extraer su mensaje. Una vez se ha extraído el mensaje, se podrá comprobar si está cifrado o no buscando en él alguno de los caracteres especiales que lo identifiquen como tal. La librería que se utilizará será la ZXing porque, al contrario que en el módulo de la cámara, esta librería puede realizar este proceso de una forma rápida y óptima sin que para esto se necesite que el usuario tenga instalada su aplicación principal en el terminal.

Además, si por algún casual, el explorador de archivos dejase de responder, la aplicación devolverá al usuario a la pantalla principal y mostrará un mensaje de error.

#### 5.4.2.2 DESCIFRAR EL MENSAJE

En este subproceso, la aplicación llamará al proceso de descifrado que ya se ha diseñado previamente en el módulo de lectura por cámara. Este proceso actuará de la misma forma ya que la información que le llega desde la cámara y desde el fichero es la misma: el mensaje cifrado.

#### 5.4.2.3 REDIRECCIONAR URL

Al igual que en el subproceso anterior, esta etapa coincide con el subproceso homónimo diseñado en el módulo de lectura por cámara. Con ánimo de no repetir más veces lo mismo, se obviará el diseño de esta tarea.

#### 5.4.2.4 MOSTRAR MENSAJE

De la misma forma que en los subprocesos anteriores, esta etapa realiza exactamente lo mismo que la etapa con el mismo nombre del módulo de lectura por cámara. Por las mismas razones, no se diseñará.

### 5.4.3 MÓDULO DE ACTUALIZACIÓN

Este módulo es muy simple. El usuario debería poder consultar en todo momento las novedades que se han implementado en la versión de la aplicación que tiene instalada en su terminal. Para ello, el usuario tendrá que abrir el menú deslizable donde se encuentran todas las opciones (que se diseñará más adelante) y pulsar sobre un botón con la etiqueta "Novedades de la versión". La aplicación mostrará un diálogo al usuario por pantalla con el log que contiene toda la información que éste busca.

### 5.4.4 MÓDULO DE CAMBIO DE AJUSTES

Al igual que el anterior, este módulo es mucho más simple que los de generación y lectura. Su funcionalidad consiste en permitir que el usuario modifique algunas opciones que tendrá la aplicación configuradas por defecto. Estas opciones serán la

ruta de la carpeta de salida donde se guardarán los códigos QR y QRypt que se generen con la aplicación y el idioma de la aplicación.

En el caso de elegir modificar la ruta de la carpeta de salida, aparecerá un diálogo en el que el usuario puede introducir la ruta que quiera. Cuando se introduzca, la aplicación realizará algunas comprobaciones para validar la ruta. Las comprobaciones serán:

- La ruta pertenece a un lugar real del sistema de archivos del terminal.
- La ruta no contiene caracteres no soportados.

Por otro lado, si se selecciona modificar el ajuste de idioma, la aplicación mostrará un diálogo con los idiomas soportados (en esta versión serán *Español* e *Inglés*) y el usuario tendrá que elegir cuál prefiere.

## 5.5 INTERFACES DE USUARIO

Uno de los puntos más importantes de la aplicación ya que figura también en los requisitos de usuario es que el diseño de la interfaz de usuario debe ser simple y accesible para personas con problemas de daltonismo. A continuación, se diseñarán las distintas interfaces del sistema.

### 5.5.1 ESTUDIO DE DISEÑO

Antes de empezar a diseñar, es preciso realizar un estudio para comprender los problemas y posibles soluciones que surgen con los requisitos de usuario que se refieren al diseño de las interfaces.

#### *5.5.1.1 SENCILLEZ E INTUICIÓN*

El requisito de usuario RUC-21 declara que la interfaz deberá ser sencilla e intuitiva. Para saber qué hacer antes de empezar con el diseño, se debe entender a qué se le llama sencillo y a qué intuitivo:

- **Sencilla:** Que no ofrece dificultad. La interfaz debe ser lo suficientemente simple como para que cualquier usuario pueda aprender a usar las principales funciones de la aplicación sin ningún problema ni dificultad. [6]
- **Intuitiva:** Que permite su comprensión instantáneamente, sin necesidad de razonamiento. La navegación por la interfaz debe coincidir con los movimientos táctiles que realizaría en esa situación la media de los usuarios. [5]

Para que sea sencilla, se llenará con el mínimo número de opciones y botones la pantalla. De esta forma, el usuario no tendrá que buscar por toda la interfaz la opción o el botón que permite ejecutar la función que él desea.

Por otro lado, para que se cumpla el requisito de intuición, se seguirán las recomendaciones de diseño oficiales del equipo de Google. Estas recomendaciones son las siguientes:

- El usuario debe saber en cualquier momento lo que está haciendo la aplicación. Si el usuario pulsa un botón o una opción, ésta se resaltará para hacer saber al usuario que la aplicación ha detectado la acción de su dedo. [8]
- Usar colores primarios. Se utilizarán los colores oficiales de la paleta que el equipo de Google ha puesto a la disposición de aquellos que la necesiten. [10]

Ilustración 19: Paleta de colores oficial del equipo de Android

- Utilizar iconos para simbolizar lo que se quiere decir. Los usuarios generalmente se fijan antes en un icono que en un texto. Los iconos deben figurar con bisel para que hagan ilusión de relieve. Los iconos que se utilizarán

serán de la base de datos oficial de iconos Android la no ser que no exista uno que represente lo que se quiere decir.[11]
- Utilizar un estilo de escritura conciso, simple, amigable, con lo más importante al principio, sin repeticiones y describiendo sólo lo principalmente necesario. [12]
- Utilizar un tema que cumpla los estándares de Holo. Estos temas tienen barra superior (llamada *ActionBar*) con opciones. Además, su navegación se basa mucho en los gestos. [7]
- Utilizar la tipografía oficial de Android llamada *Roboto*. [9]

#### 5.5.1.2 DALTONISMO

Existe otro requisito de usuario que se refiere al diseño de las interfaces. Este es el RUC-22, que exige que la aplicación debe ser accesible para las personas que padezcan daltonismo.

Para poder plantar cara al problema, se debe conocer el significado de los conceptos básicos que se relacionan con este problema de accesibilidad.

El término daltonismo significa "Defecto de la vista que consiste en no percibir determinados colores o en confundir algunos de los que se perciben" según la RAE. [14]

Para intentar eludir este problema y cumplir con el requisito, es menester conocer más detalles sobre esta enfermedad. Por ejemplo, ¿qué colores son los que no se perciben o provocan alguna confusión? ¿Qué medidas se deben tomar para que la aplicación sea accesible para personas con esta deficiencia?

Se intentará responder a estas preguntas sin ahondar en los términos muy especializados, ya que son muchos y no hace falta conocerlos para solventar el problema.

Se sabe que existen 3 tipos de daltonismo: [13]

- **Acromático:** Las personas que lo sufren son incapaces de detectar ningún color. Sólo perciben el color en escala de grises.
- **Dicromático:** Las personas que lo padecen son incapaces de ver el rojo, el verde o el azul.
- **Tricromático:** Las personas que lo padecen tienen dificultades para ver el rojo, el verde o el azul. Es el más padecido por la población.

Para evitar que las personas que padezcan daltonismo tengan algún problema con la aplicación y siguiendo la guía de accesibilidad de W3C, se tendrán en cuenta las siguientes recomendaciones:

- Los textos y gráficos serán comprensibles cuando se vean sin color. El color no se usará por sí mismo para transmitir información. [15]

- El color no se usará como único medio visual para transmitir la información, indicar una acción, solicitar una respuesta o distinguir un elemento visual. El color se utilizará como apoyo para resaltar dicha información. [16]
- No seleccionar colores problemáticos para los elementos de navegación. Los elementos de navegación son de importancia crítica en una aplicación; por lo tanto, se debe pensar con detenimiento su diseño con el fin de evitar estos problemas. [13]

Centrándonos en los tipos de daltonismo que influyen a un color y no a todos, básicamente y a grandes rasgos, los daltónicos tienen problemas con los siguientes colores: Naranja, Rojo, Verde, Azul y Violeta. Ellos dividen estos colores en 2 grupos:

1. Naranja, Rojo y Verde.
2. Azul y Violeta.

Dos colores que pertenecen al mismo grupo son, para ellos, los mismos.

Ilustración 20: Ejemplo de daltonismo 1

En la Ilustración 20, se muestran dos fotos. La de la izquierda representa cómo ve ese objeto una persona sin daltonismo, la de la derecha representa cómo lo ve una persona con daltonismo. Como se puede observar, los colores de tonalidad naranja, roja o verde, se representan casi como un color parecido al marrón.

Ilustración 21: Ejemplo de daltonismo 2

En la Ilustración 21, se muestran otras dos fotos. Otra vez, la de la izquierda representa cómo ve ese objeto una persona sin daltonismo y la de la derecha representa cómo lo ve una persona con daltonismo. Como se puede observar, los colores de tonalidad azul o violeta, se representan casi como un color parecido al azul.

La solución para estos casos de daltonismo es la elección de sólo un color de cada grupo de colores problemáticos a la vez. De esta forma, nunca podrán confundir dos colores (al no ser que su daltonismo sea acromático). [17]

Entendidos los conceptos y comprendido el problema, se puede concluir con que el diseño de la interfaz debe ser completamente representativo sin colores, debe utilizar los colores sólo como apoyo o adorno, y debe contener sólo uno de los colores de cada grupo de colores problemáticos para los daltónicos a la vez.

#### *5.5.1.3 VOLVER ATRÁS*

Según el requisito de usuario RUC-26, la aplicación debería ser capaz de volver a la primera actividad desde cualquier actividad únicamente pulsando un botón.

Para ello, se pondrá a disposición del usuario un botón en la parte superior izquierda de la pantalla que permita volver a la pantalla principal desde cualquier actividad. Además, al pulsar el botón "atrás" del dispositivo móvil se logrará el mismo objetivo.

#### 5.5.1.4 PANEL DE CONTROL DESLIZABLE

El último requisito de usuario que se refiere de algún modo al diseño de la interfaz es el RUC-25. Este requisito exige que el usuario será capaz de acceder a un panel de control deslizable para cambiar los ajustes desde cualquier ventana de la aplicación.

Para ello, se diseñará un menú deslizable con las opciones que se podrán modificar. Este menú aparecerá en la pantalla desde cualquier actividad deslizando el dedo hacia la derecha o pulsando el botón "menú" del terminal móvil.

### 5.5.2 PANTALLA PRINCIPAL

La pantalla principal estará compuesta por 2 interfaces distintas dependiendo de si lo que se quiere es generar o leer un código QR o QRypt.

#### 5.5.2.1 INTERFAZ DE GENERACIÓN

La interfaz de generación mostrará 2 botones: "Generar QR" y "Generar QRypt", los cuáles permitirán al usuario ejecutar los módulos de generación de códigos QR o QRypt, respectivamente.


Ilustración 22: Interfaz de usuario de la pantalla principal 1

Como se puede observar en la Ilustración 22 el diseño es muy simple, lo que produce que la aplicación sea muy fácil de usar. Además, los colores utilizados pertenecen a las escalas de blancos, negros y azules; por lo tanto, el diseño de la aplicación no generaría confusión en gente que sufra problemas de daltonismo.

#### 5.5.2.2 INTERFAZ DE LECTURA

La interfaz de lectura mostrará también 2 botones: "Leer de cámara" y "Leer de fichero", los cuáles permitirán al usuario ejecutar los módulos de lectura por cámara y lectura por fichero de códigos QR y QRypt, respectivamente.


Ilustración 23: Interfaz de usuario de la pantalla principal 2

Como se muestra en la Ilustración 23, el diseño de esta interfaz es casi igual que el de la anterior, simplemente cambiarán los botones y la imagen del código QR en la parte central superior de la pantalla.

Las formas de navegar entre una interfaz y otra de la pantalla principal serán muy simples:

- Pulsar en la barra superior la interfaz ("Generar" o "Leer") que se desea.
- Deslizar el dedo hacia la izquierda o derecha.

- Si se está en la interfaz de generación y se desliza el dedo a la izquierda, se cambiará la interfaz.
- Si se está en la interfaz de lectura y se desliza el dedo a la derecha, se cambiará la interfaz.

Como se puede ver en la barra superior, la interfaz que se esté viendo en el momento aparecerá subrayada.

### 5.5.3 PANTALLA DE GENERACIÓN DE CÓDIGO QR

Esta pantalla seguirá la misma filosofía de color que la anterior. Lo que se busca con esta pantalla es permitir que el usuario introduzca el mensaje que quiere incluir en un código QR simple y generarlo.

#### *5.5.3.1 INTERFAZ DE ESCRITURA DEL MENSAJE*

Así que, la interfaz mostrará un campo donde introducir el mensaje y un botón situado en la barra superior para hacer saber al programa que el mensaje está introducido.


Ilustración 24: Interfaz de usuario de la pantalla de escritura del mensaje

Mientras el usuario introduzca el mensaje, el 0 situado debajo del campo "mensaje" y a la izquierda del 2000, empezará a incrementarse en tiempo real reflejando los

caracteres escritos hasta el momento. El 2000 representa los caracteres máximos que la aplicación permitirá escribir.

Además, en la barra superior, aparecerá un icono con la palabra "QRypt" al lado (el nombre de la aplicación) que, si se pulsa (el icono), conducirá al usuario de vuelta a la pantalla principal. Esto también puede hacerse con el botón "atrás" del dispositivo.

#### 5.5.3.2 INTERFAZ DE "MOSTRAR Y COMPARTIR EL CÓDIGO"

Una vez se escriba el mensaje y se pulse el botón situado en la esquina superior derecha, la aplicación generará el código QR y cambiará a la interfaz de "Mostrar y compartir el código".


Ilustración 25: Interfaz de usuario de la pantalla "Mostrar y compartir el código"

Esta interfaz contará con, además del botón de vuelta atrás que coincide con el icono situado en la parte izquierda de la barra superior, una imagen previa del código QR generado con el mensaje que ha generado el usuario y dos botones, situados en la parte derecha de la barra superior, para elegir la aplicación a la que se desea compartir el código QR mediante un menú desplegable o elegir mandarlo directamente a la última aplicación que se usó para este fin (en este caso Dropbox).

### 5.5.4 PANTALLA DE GENERACIÓN DE CÓDIGOS QRYPT

Las interfaces de este módulo son iguales que las del módulo de generación QR salvo que, en este caso, cuenta con más pantallas: la de selección de tipo de cifrado, selección de contraseña y selección de color.

Las 3 pantallas nuevas son diálogos emergentes que muestran al usuario las opciones que pueden escoger y que cuentan con 2 botones en la parte inferior: "Aceptar" y "Cancelar". Si se pulsa "Aceptar" se navegará hasta la siguiente pantalla. Si se pulsa "Cancelar" se volverá a la pantalla de escritura del mensaje.

#### *5.5.4.1 INTERFAZ DE SELECCIÓN DE TIPO DE CIFRADO*

Una vez se ha introducido un mensaje para cifrar, aparecerá un diálogo emergente en el que aparecerán los distintos tipos de cifrado que pueden aplicarse, permitiendo que el usuario escoja uno o más de éstos.


Ilustración 26: Interfaz de la pantalla de selección del tipo de cifrado

Como se puede observar en la Ilustración 26, las opciones que se ofrecerán en este diálogo emergente se presentarán en forma de cuadrícula. Además, el diálogo contará con 2 botones, uno para volver a la pantalla de creación del mensaje y otro para terminar de elegir el tipo de cifrado y continuar a la siguiente pantalla.


Ilustración 27: Interfaz de la pantalla de selección del tipo de cifrado 2

En la Ilustración 27 se muestra cómo sería la interfaz si se eligiese cifrar el mensaje por medio de una fecha y de un patrón. Como se puede ver, el programa permitiría que el usuario pueda cifrar con más de una de las opciones. El contorno de las imágenes se resaltaría en azul para que el usuario sepa qué opciones están señaladas en ese momento y cuáles no.

#### 5.5.4.2 INTERFAZ DE ESCRITURA DE CONTRASEÑA

En el caso de que se haya elegido la opción de contraseña como tipo de cifrado, la aplicación exigirá al usuario que escriba una mediante otro diálogo.


Ilustración 28: Interfaz de la pantalla de escritura de contraseña

El diálogo emergente contará con los botones que cuentan todos los diálogos de esta sección y con un título, un subtítulo y un campo para escribir la contraseña que se usará para cifrar el mensaje introducido.

#### *5.5.4.3 INTERFAZ DE SELECCIÓN DE FECHA*

En el caso de que se haya elegido la opción de fecha como tipo de cifrado, la aplicación exigirá al usuario que seleccione una fecha exacta en un marcador mediante otro diálogo.

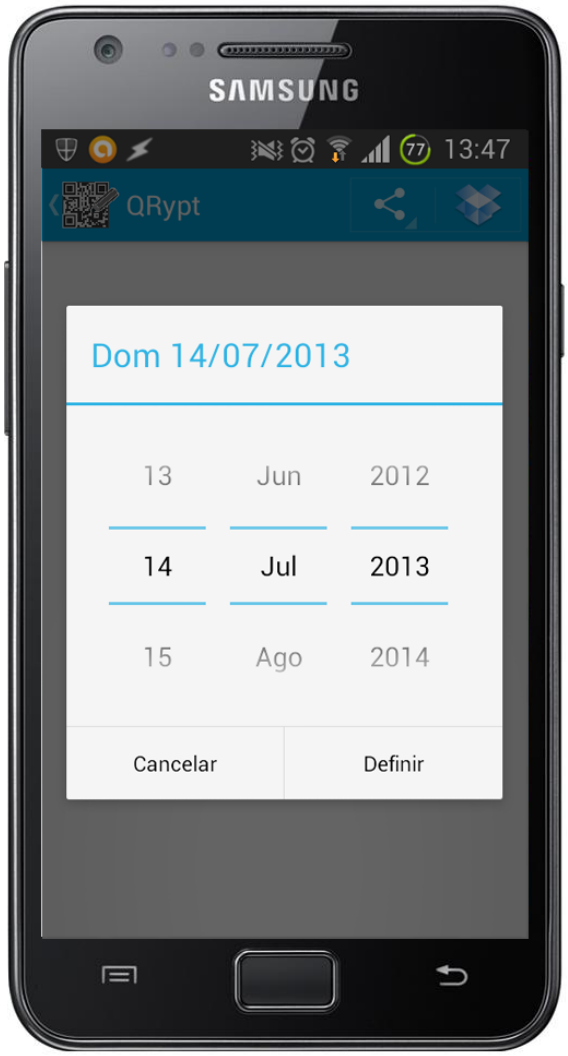



Ilustración 29: Interfaz de la pantalla de selección de fecha

El marcador (traducción de la clase *Dialer* de Android) mostrará en el título del diálogo la fecha del día en que se intenta generar el código QRypt y 3 columnas que representan un día del año. Estas columnas pueden cambiar su valor bien escribiéndolo en ellas o bien deslizando el dedo para que se seleccionen valores siguientes o anteriores. La fecha que se elija en esta pantalla será la fecha a partir de la cual se permitirá leer el código QRypt. Hasta que no se cumple la fecha especificada, el código permanecerá bloqueado.

#### 5.5.4.4 INTERFAZ DE SELECCIÓN DE PATRÓN

En el caso de que se haya elegido la opción de patrón como tipo de cifrado, la aplicación exigirá al usuario que introduzca un patrón de desbloqueo en un diálogo diseñado explícitamente para esa función.


Ilustración 30: Interfaz de la pantalla de selección de patrón 1

En el diálogo aparecerán 9 puntos con los que se debe generar un patrón de bloqueo y desbloqueo deslizando el dedo sobre ellos. Los puntos elegidos para el patrón se resaltarán por encima de los demás y una línea recta unirá los puntos en el orden en el que se eligieron.


Ilustración 31: Interfaz de la pantalla de selección de patrón 2

Lo que se muestra en la Ilustración 30 son los puntos cuando aun no se ha elegido un patrón como contraseña para cifrar el mensaje, mientras que, lo que se muestra en la Ilustración 31, es cuando ya se ha elegido un patrón. Este patrón pude corresponderse con la contraseña "*3214789*" o con "*9874123*".

#### *5.5.4.5 INTERFAZ DE SELECCIÓN DE COLOR*

Una vez se ha elegido el tipo de cifrado y se ha especificado la contraseña, la fecha o el patrón, se elige el color. Para ello y siguiendo la misma filosofía que en las demás interfaces de este módulo, se hará uso de un diálogo emergente para elegir el color. Se usará una arquitectura de tipo matricial en vez de una lista, al igual que se hizo en la interfaz de elección del tipo de cifrado, pero en esta no se podrá elegir más de un color.

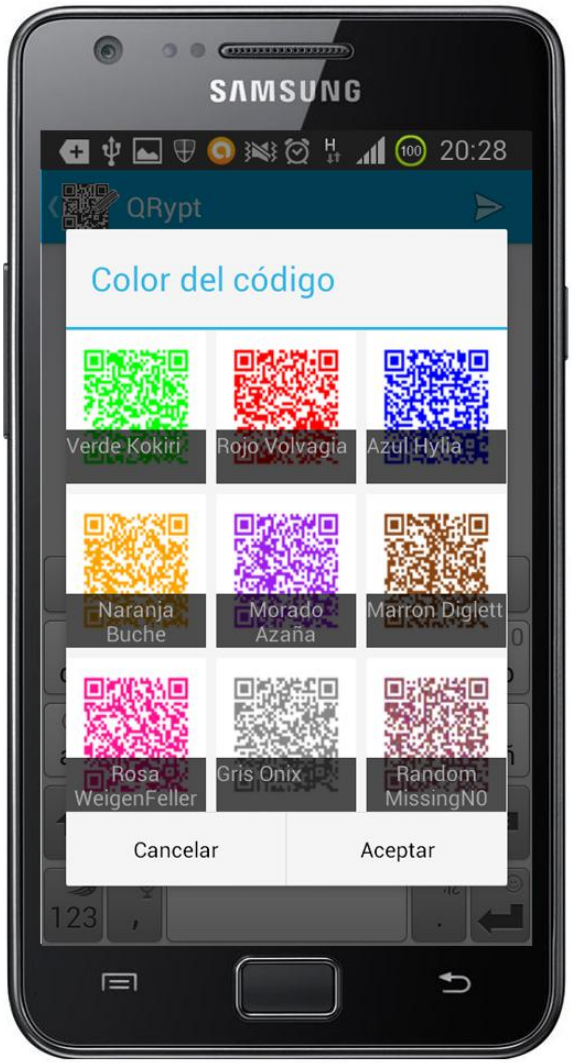



Ilustración 32: Interfaz de la pantalla de selección de color 1

Como se puede ver en la Ilustración 32, se adornará el nombre de los colores con un adjetivo de naturaleza graciosa, ya sea haciendo alusión a términos de videojuegos (Zelda, Pokemon) como a nombres de personas reales (Azaña, Weigenfeller). El color "Random MissingN0" consiste en elegir colores de forma aleatoria que no entren en el umbral de lo que se considera blanco para formar un código de apariencia extraña. Cada color aleatorio que se genera se utiliza para un solo pixel.

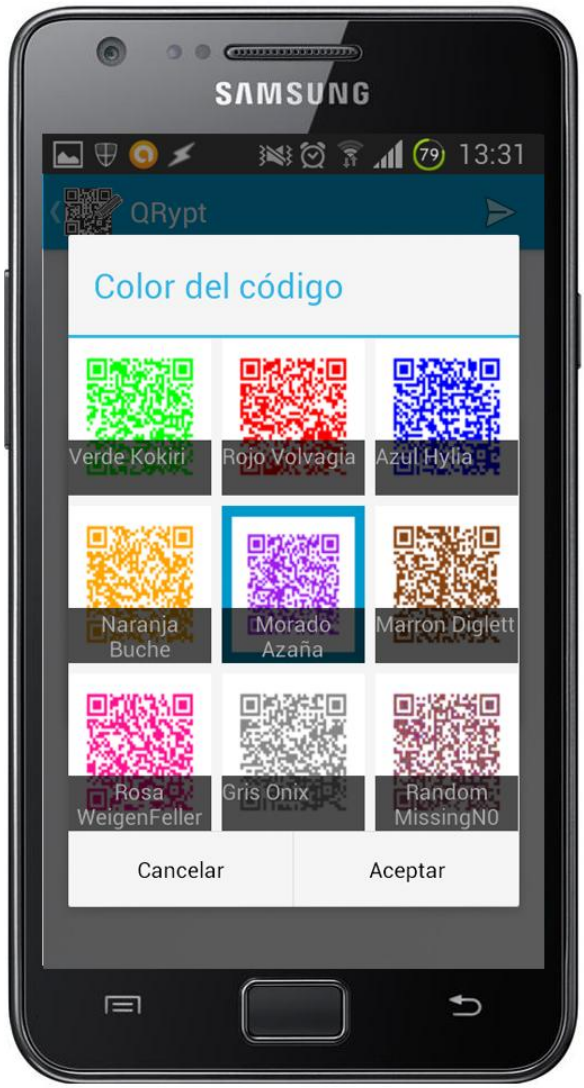



Ilustración 33: Interfaz de la pantalla de selección de color 2

Como se diseñó en la interfaz de elección de tipo de cifrado para el mensaje y como se puede observar en la Ilustración 33, cuando se elija un color en esta interfaz, se marcará con un borde azul alrededor para que el usuario sea consciente del color que ha elegido.

La elección del color del código QRypt que se va a generar será la última opción en la que se consultará al usuario. Después de elegir el color, la aplicación generará el código QRypt mostrándolo de la misma forma y en la misma interfaz que la de "mostrar y compartir el código" del módulo de generación QR.

### 5.5.5 PANTALLA DE LECTURA DE CÓDIGOS POR CÁMARA

Las interfaces de este módulo son, principalmente 2: la interfaz de la cámara y la interfaz en la que se muestra el mensaje. Aun así, si se diese el caso de que hubiese que introducir una contraseña o un patrón porque el mensaje del código está cifrado con alguna de estas opciones, surgirían los diálogos de introducción de contraseña y de patrón correspondientes. Como las interfaces de estos últimos diálogos ya está

diseñadas porque son idénticas a las que se utilizan en el módulo de generación de códigos QRypt, en esta sección se obviarán.

#### 5.5.5.1 INTERFAZ DE LA CÁMARA

La interfaz de la cámara será muy simple. Contará con una barra superior con el botón de volver atrás (para seguir con la filosofía de diseño de la aplicación), con un cuadro grande en el que se mostrará lo que se está enfocando por la cámara, y un botón de "Escanear" por si acaso la cámara ha tenido algún error y se ha congelado.

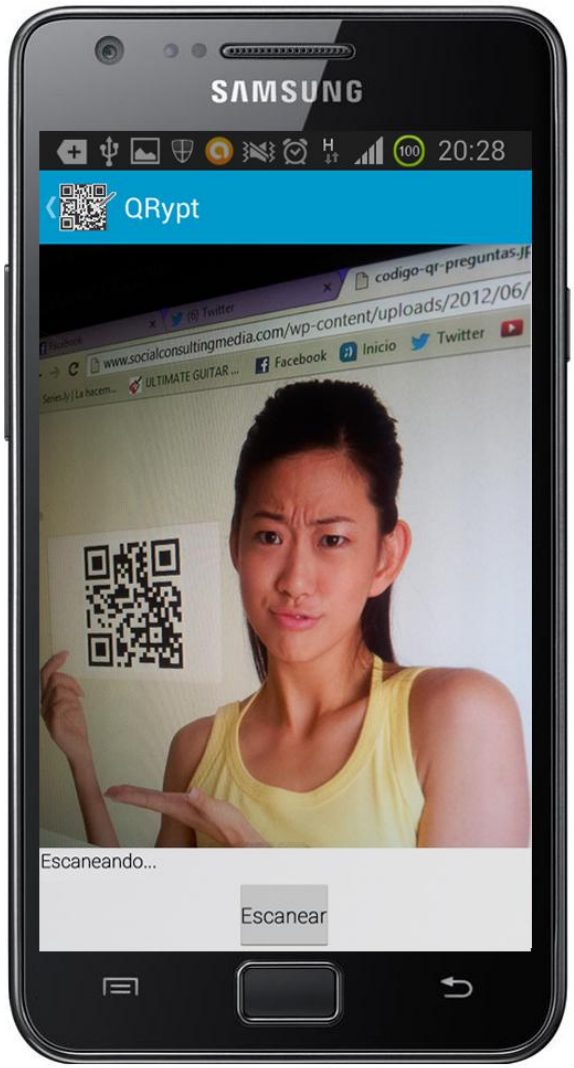



Ilustración 34: Interfaz de la pantalla de cámara

El cuadro donde saldrá la imagen que se está enfocando en tiempo real con la cámara del dispositivo tendrá que ser grande ya que es lo más importante de esta interfaz. El usuario tiene que saber a qué lugar está apuntando en todo momento con el máximo posible de detalles. Si se pulsase el botón de arriba a la izquierda o el botón "atrás" del dispositivo Android, la aplicación volvería a la pantalla principal.

#### 5.5.5.2 INTERFAZ DE LA PANTALLA DE "MOSTRAR EL MENSAJE"

Si al leer el código QR o QRypt, se identifica que el mensaje que está incluido en éste está cifrado, la aplicación intentará descifrarlo por sus propios métodos al no ser que el cifrado sea de tipo contraseña o patrón. Si el cifrado utilizado pertenece a alguno de estos dos tipos, la aplicación mostrará al usuario los diálogos de escritura de contraseña y patrón correspondientes. Las interfaces de estos diálogos serán las mismas que las utilizadas en el módulo de generación de códigos QRypt.

Una vez introducidos los datos y descifrado el mensaje con éxito, se mostrará éste. La interfaz de la pantalla donde se mostrará será muy simple. Se mostrará el clásico botón que se ha utilizado en casi todas las interfaces de "volver atrás", y se mostrará el mensaje incluido en el código bajo un título en el que pondrá "Mensaje".


Ilustración 35: Interfaz de la pantalla de "Mostrar el mensaje descifrado"

### 5.5.6 PANTALLA DE LECTURA DE CÓDIGOS POR FICHERO

Las interfaces que complementarán este módulo son idénticas a las que componen el módulo de lectura de códigos por cámara salvo que este módulo ejecutará el explorador de archivos del terminal. Como este explorador es una aplicación de terceros y no se va a diseñar, no se incluye en este proyecto. Además, los diálogos

que aparecerán a la hora de descifrar un mensaje y la pantalla en la que se muestra éste comparten la misma interfaz que estas mismas pantallas en el módulo de lectura de códigos por cámara; por lo tanto, se obviará su inclusión en este apartado.

Como, por otro lado, la aplicación debería ser capaz de descifrar códigos QR y QRypt que se encuentren en imágenes que envíen desde otras aplicaciones, es necesario diseñar cómo aparecerá en el menú "Compartir con" de las aplicaciones Android que pueden enviar imágenes a otras aplicaciones.

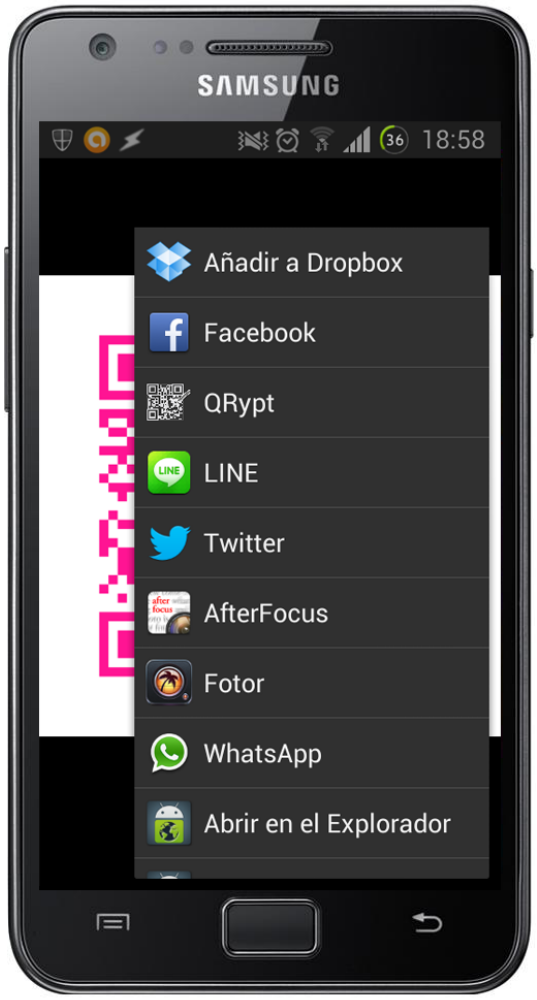



Ilustración 36: Interfaz del menú desplegable "Compartir con" de aplicaciones de terceros

Como se puede ver en la Ilustración 36, desde la galería de imágenes podría enviarse una cualquiera a la aplicación que se está desarrollando. La aplicación aparece en el tercer puesto pero, si se usara con más asiduidad, aparecería en un puesto superior. Si se eligiese la aplicación *QRypt*, el programa intentaría descifrar el mensaje que contiene directamente.

### 5.5.7 MENÚ DESPLEGABLE

El menú desplegable contendrá las opciones básicas que se pueden cambiar desde la aplicación, además de la información básica de la aplicación y sus versiones. Este

menú aparecerá en la pantalla si se hace un gesto deslizando el dedo hacia la derecha o pulsando el botón "menú" del terminal.


Ilustración 37: Interfaz del menú desplegable

El menú aparece de la parte izquierda de la pantalla arrastrando con él la pantalla que estuviera viéndose en ese momento. Las opciones que ofrece a modo de vista son cambiar la carpeta de salida donde se guardarán los códigos generados con la aplicación, cambiar el idioma y acceder a un menú adicional en el que se cambiarán las opciones básicas de cifrado como elegir el color por defecto, la contraseña, etc. Esta última opción se ha diseñado con vistas al futuro ya que no se implementará en esta versión de la aplicación. Además de las opciones que ofrecerá, también permitirá acceder a información sobre la aplicación como las novedades que se han incluido en la versión actual frente a las demás, información general sobre la aplicación en la que se incluirá el nombre de su desarrollador y sobre qué trata la aplicación, y una opción para comprobar actualizaciones en tiempo real que tampoco se implementará en esta versión.

#### 5.5.7.1 INTERFAZ DE CAMBIO DE RUTA DE LA CARPETA DE SALIDA

Una vez se pulse en la opción de carpeta de salida, aparecerá un diálogo emergente en el que se mostrará la ruta actual de la carpeta de salida y en el que se podrá modificar ésta.

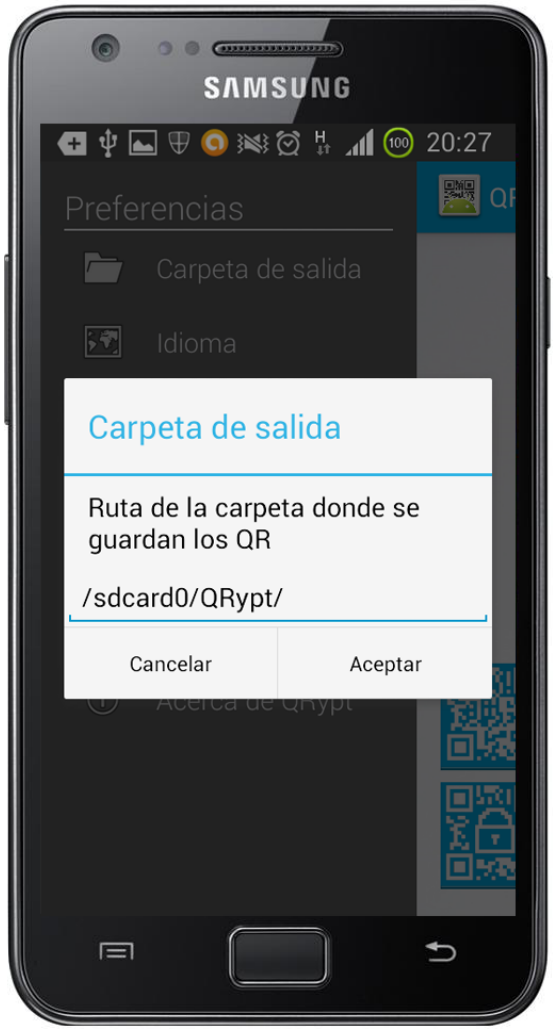



Ilustración 38: Interfaz de la pantalla de modificación de la ruta de la carpeta de salida

Como en todos los diálogos emergentes diseñados hasta ahora, se cuenta con 2 botones inferiores para guardar los cambios ocasionados en la ruta de la carpeta de salida u omitirlos.

#### 5.5.7.2 INTERFAZ DE SELECCIÓN DE IDIOMA

Una vez se pulse en la opción de modificar el idioma de la aplicación, aparecerá un diálogo emergente en el que se expondrá unos botones radiales (de la clase *RadioButton* de Android) que darán a elegir al usuario el idioma en el que quiere que se muestre la aplicación.

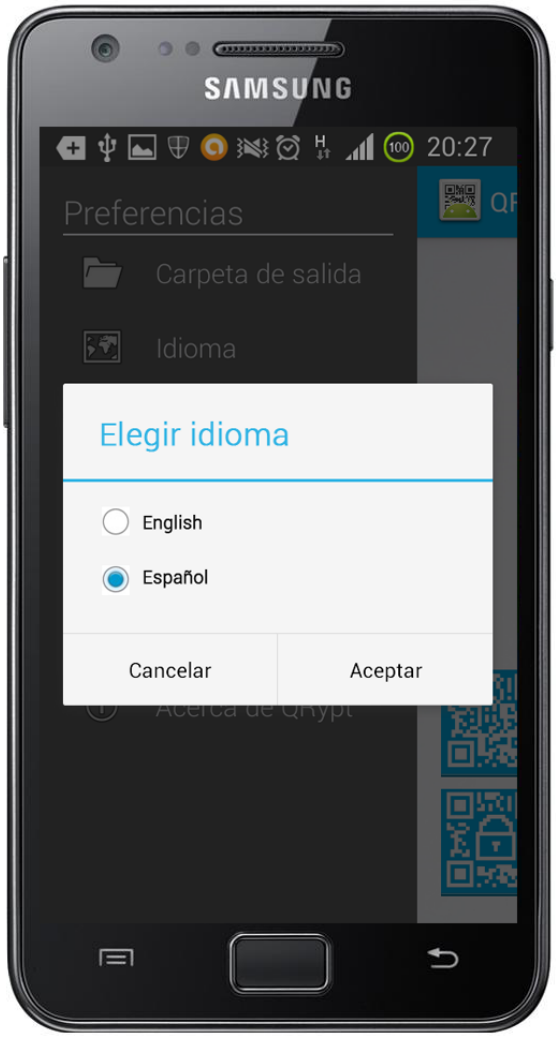



Ilustración 39: Interfaz de la pantalla de modificación del idioma

Al igual que en el diálogo emergentes anterior, se cuenta con 2 botones inferiores para guardar los cambios u omitirlos.

#### 5.5.7.3 INTERFAZ DE NOVEDADES DE LA VERSIÓN

Una vez se pulse en la opción de mostrar las novedades de versión, aparecerá un diálogo emergente en el que se expondrá el número de versión que está instalada en el terminal y las novedades que impuso esta versión frente a las anteriores.


Ilustración 40: Interfaz de la pantalla de novedades de la versión

A diferencia de los demás diálogos, éste no cuenta con 2 botones. En este diálogo no se puede modificar nada; por lo tanto, no tiene sentido tener botones que omitan o guarden los cambios que produzca el usuario.

# 6 IMPLEMENTACIÓN

Esta sección pretende emprender la implementación del sistema de información que se ha analizado previamente en el Análisis y se ha diseñado en el Diseño. Además, se justificarán todas y cada una de las decisiones que se tomen.

## 6.1 MÓDULO DE GENERACIÓN DE CÓDIGOS QR A PARTIR DE UN MENSAJE

El método principal para generar un QR recibe 3 atributos: el mensaje que se va a introducir en el código, el tipo de código que es (QR o QRypt) y el color en el que se va a generar éste.

Para generar el código se hace uso de una librería para Java que es capaz de lograr este objetivo y, además, pertenece a la aplicación que es la más utilizada en Android para tratar códigos de naturaleza QR. Esta librería se llama ZXing.

### 6.1.1 ZXING

Se pronuncia "Zebra Crossing" y es una librería de código abierto para procesar imágenes con códigos de barras multi-formato de una o dos dimensiones. Está implementada en Java. El objetivo de sus desarrolladores es el usarla a través de las cámaras de los dispositivos móviles actuales para escanear códigos de barras sin comunicarse con un servidor.

Lo bueno de esta librería es que permite realizar exactamente lo que se intenta desarrollar en este punto pero lo malo es que está escrita en Java y contiene clases que no existen de Android. Este problema se soluciona si se instala su aplicación llamada "Barcode Scanner" de lectura de códigos QR. De esta forma, cuando se va a generar un código o se va a escanear, se llama a su aplicación y ésta ya lo hace y envía la información de vuelta a la aplicación que se está desarrollando. No se contempla la opción de obligar a los usuarios a instalar la aplicación "Barcode Scanner" ya que no se consideraría una buena competencia si se depende de una aplicación de terceros para realizar lo básico. Debido a esto, se procederá a adaptar la librería para poder utilizarla en Android.

Esta librería, al generar un código QR, primero transforma un mensaje en una matriz de bits y, después, pasa esta matriz a una imagen interpretando los bits de la matriz. A la hora de transformar el mensaje en matriz no hay ningún problema pero, cuando va a dibujar la matriz, hace uso de la clase de Java "*BufferedImage*" y ésta no existe en el dominio de Android.

Para solventar este problema, se ha decidido modificar el código para que, en vez de que llame a la clase "*BufferedImage*", haga uso de los objetos "*Bitmap*" que son los nativos de Android para manejar imágenes. Por suerte, la forma de manejar uno u otro de estos objetos es muy similar. La librería ZXing, al usar "*BufferedImage*", se recorre todas las posiciones de la matriz y dibuja cada pixel de la imagen utilizando el método *setRGB*. Además, la clase *BufferedImage* permite dar formato a la imagen directamente. Con la clase *Bitmap* se escribirán los pixeles de la imagen utilizando el

método setPixel y habrá que crear un objeto *File* antes de darle formato con el método *compress* de la clase *Bitmap*.

### 6.1.2 DRAWQR

Ya se ha hablado del método *drawQR* que es el que se utiliza para generar códigos de naturaleza QR, pero no se ha ahondado en su algoritmo. Hace falta recordar que este método recoge como atributos el mensaje que se va a introducir en el código, el tipo de código que es (QR o QRypt) y el color en el que se va a generar éste. El algoritmo del método, de forma simplificada, es como se muestra a continuación.

drawQR(mensaje, tipo, color):

- Se instancia un objeto QRCodeWriter de la librería ZXing.
- Se transforma el mensaje a matriz de bits mediante el método *encode* del objeto QRCodeWriter.
- Para cada posición de la matriz:
  - Si existe un valor, se dibuja un pixel en el *Bitmap* del color elegido con el método *setPixel*.
  - Si no, se dibuja un pixel en el *Bitmap* de color blanco con el método *setPixel*.
- Se obtiene la ruta de la carpeta de salida y se crea un objeto *File* vinculado a ella.
- Se obtienen los segundos, minutos, horas, día, mes y año actuales para crear el nombre del archivo.
- Se crea el archivo y se le da formato .PNG mediante el método *compress* de la clase *Bitmap* a partir del Bitmap con el código QR y se vincula al archivo creado.
- Se devuelve el Bitmpa creado para representarlo en un objeto *ImageView* para que el usuario pueda verlo desde la aplicación.

**Tabla 119: Algoritmo del método *drawQR***

Como se puede observar, se hace uso de la librería únicamente para transformar el mensaje en matriz de bits ya que, como se ha explicado antes, esta librería hace uso de clases de Java que no existen en Android. A pesar de eso, la importancia de el método de esta librería es vital ya que permite generar la matriz que se dibujará como el código QR sin necesidad de conectarse a Internet.

## 6.2 MÓDULO DE CIFRADO DE MENSAJES

Como se explicó en el diseño del sistema, la aplicación hará uso de unos algoritmos de cifrado que se emplearán en el mensaje para obtener una versión protegida de éste. En este capítulo se van a explicar un poco más en profundidad estos algoritmos.

Como se pudo ver en la Ilustración 13, el proceso de seguridad que se va a implementar empieza cifrando el mensaje, luego lo desordena y, finalmente, lo codifica en *base64* e incluye los tokens de control.

### 6.2.1 CIFRAR

Este método se utiliza para cifrar el mensaje a partir de una contraseña; por lo tanto, recibe como atributos un mensaje y una contraseña. Su algoritmo, en muy alto nivel, es como se muestra a continuación.

cifrar(mensaje, contraseña):

- Se genera un *salt* aleatorio:
  - Se inicializa un objeto *SecureRandom* y se genera un *salt*.
  - Se genera el hash del *salt* utilizando la función resumen SHA-1.
  - Se concatena el hash generado con la contraseña.
- Se deriva la contraseña:
  - Se genera un número aleatorio entre 1000 y 11000 que será el número de iteraciones en la función de derivación de la contraseña.
  - Se deriva la contraseña haciendo uso de la función de derivación "PBKDF2WithHmacSHA1" tantas veces como represente el número de iteraciones aleatorio que se ha obtenido anteriormente.
- Se cifra el mensaje:
  - Se genera un vector de inicialización aleatorio.
  - Se cifra el mensaje con la clave obtenida anteriormente y el vector de inicialización utilizando el algoritmo de cifrado AES.
- Se devuelve el mensaje cifrado.

Tabla 120: Algoritmo del método *cifrar*

En la tabla anterior se observa que el algoritmo de cifrado es idéntico al que se diseñó en la sección de diseño del sistema y que figura en la Ilustración 14. A pesar de que en el Estado del Arte se explicaba cada uno de los conceptos de cifrado que se iban a utilizar en los algoritmos, a continuación se expone cómo y por qué se utiliza cada uno.

#### 6.2.1.1 CONTRASEÑA

Cuando se habla de la contraseña que recibe el método, no es otra que el conjunto de valores tomados a la hora de seleccionar los tipos de cifrado que se desean utilizar para cifrar el mensaje. Una vez se tienen los valores, se genera un hash de cada uno utilizando la función resumen SHA-1 y se concatenan. El valor correspondiente es el que se usa como atributo de entrada en este método.

#### 6.2.1.2 SALT

En el algoritmo, el *salt* se genera aleatoriamente utilizando la clase *SecureRandom*. Se utiliza la clase *SecureRandom* y no *Random* porque el número que genera el primero es criptográficamente más fuerte. Más tarde se genera un hash del *salt* utilizando la función resumen SHA-1 para mejorar la seguridad y se concatena con la contraseña.

La razón por la que se utiliza un *salt* es, simplemente, para aumentar la complejidad del proceso de seguridad. De esta forma, se hace mucho más complicada la labor de un atacante en caso de que obtuviera el criptograma.

#### 6.2.1.3 DERIVACIÓN

Se ha decidido realizar la derivación de contraseña para incrementar la dificultad para romper el criptograma debido a la carga computacional adicional que la derivación añade. La función utilizada se llama "*PBKDF2WithHmacSHA1*". Esta función recibe 4 parámetros para generar la clave que se requiere:

- La contraseña del usuario.
- El *salt* obtenido de la función de generación del punto anterior.
- El número de iteraciones para la derivación.
- La longitud de la clave que se va a obtener.

El número de iteraciones es un número aleatorio entre 1000 y 11000. Se ha elegido el número de iteraciones 1000 como mínimo debido a que:

- Requiere un tiempo de computación comedido para un terminal móvil.
- A partir de 1000 iteraciones, se generan claves suficientemente seguras y aleatorias.

Por otro lado, se ha decidido que 11000 sea el número máximo de iteraciones porque, además de la seguridad, también es muy importante la velocidad y el rendimiento del sistema. Un número mayor de iteraciones puede causar tiempos de cómputo demasiado elevados.

Por último, la longitud de la clave que se va a obtener es de 256 bits, ya que la aplicación cifra con el algoritmo AES con un tamaño de bloque de 256 bits.

#### 6.2.1.4 CIFRADO

En este algoritmo, se ha decidido utilizar el algoritmo de cifrado AES por estar considerado por la NSA como seguro. La clave de cifrado elegida es de 256 bits. Además, antes de cifrar, se generará un vector de inicialización aleatorio con tamaño de bloque AES para que, cuando se cifre el mensaje a partir de la clave generada anteriormente, los resultados del cifrado sean distintos aunque se utilizase siempre el mismo mensaje y la misma clave.

Este algoritmo de cifrado es la última tarea que se cumplirá en el proceso de cifrado de un mensaje para incluirlo en un código QRypt. Se considera un algoritmo muy seguro pero, a pesar de todo, el usuario puede mermar esta seguridad eligiendo una contraseña poco segura ya que la aplicación, por motivos de usabilidad, permite cualquier longitud de contraseña.

### 6.2.2 DESORDEN

Este método se utiliza para desordenar el mensaje cifrado a partir de una contraseña. Como siempre realiza el mismo algoritmo de desordenación para ser completamente reversible, sólo recibe un único atributo. Este atributo no es otro más que el mensaje

cifrado que tiene que desordenar. Su algoritmo, en muy alto nivel, es como se muestra a continuación.

desorden(mensaje):

- Se clona el mensaje en una variable auxiliar.
- Se declaran dos contadores: el punto de inicio del mensaje (posición 0) y el punto final (posición longitudMensaje-1).
- Se recorre todas las posiciones del mensaje salvo la última:
  - Se cambia el dato que está en la posición del mensaje por el siguiente al del contador de la posición inicial del auxiliar.
  - Se incrementa en 1 este contador de posición inicial.
  - El valor de la siguiente posición del mensaje se sobrescribe por el dato del contador de la posición final (longitudMensaje-1) del auxiliar.
  - Se resta el valor 1 al contador de la posición final.
- Si la longitud del mensaje es impar:
  - Se escribe el valor que se encuentra en el contador de la posición final del auxiliar en la penúltima posición del mensaje.
- Se devuelve el mensaje desordenado.

Tabla 121: Algoritmo del método *desorden*

Una explicación simple del algoritmo sería la siguiente: el algoritmo se encarga de recoger de forma progresiva el dato que se encuentra en la primera posición y, después, el que se encuentra en la última posición. Una vez se hace esto, se incrementa en 1 la posición del principio y se resta 1 la posición del final. El algoritmo se sigue ejecutando hasta que llega al dato que se encuentra en la posición central.

Para un mejor entendimiento del algoritmo, se añade una ilustración a continuación con el código de este método.

```
public byte[] desorden(byte[] mensaje) {
    byte[] aux = mensaje.clone();
    int puntFin = mensaje.length - 1;
    int puntIni = 0;
    for (int i = 0; i < mensaje.length - 1; i++) {
        mensaje[i] = aux[puntIni++];
        mensaje[++i] = aux[puntFin--];
    }

    if (mensaje.length % 2 != 0)
        mensaje[mensaje.length - 1] = aux[puntFin];

    return mensaje;
}
```

Ilustración 41: Captura del método *desorden*

La razón por la que se ha decidido hacer uso de este método no es otra que la de emplear una medida de seguridad adicional en el proceso de cifrado. Cuando se desordena, el mensaje queda ilegible, lo que produce que la dificultad de ataque aumente.

### 6.2.3 CODIFICACIÓN

Se ha implementado el sistema de codificación *base64*. El uso de este método se debe a que no es posible generar un código QR a partir de la estructura de bytes que se genera cuando se cifra el mensaje. Para generar un QR es necesario una estructura String y, en muchos casos, la estructura de bytes que se genera en el cifrado está compuesta por caracteres que pueden generar problemas porque no son imprimibles o no están soportados. Por eso se ha decidido utilizar la codificación *base64*. Ésta codifica cada uno de los caracteres usando 64 como base. Una vez se ha codificado, la cadena de texto generada contiene caracteres perfectamente imprimibles en el rango de caracteres alfanuméricos.

Antes de codificar el mensaje cifrado, se le concatena el *salt* empleado en la generación de la contraseña, el número de iteraciones del proceso de derivación y el vector de inicialización del proceso de cifrado. Con estos datos, será posible descifrar el mensaje después ya que, incorporado a él, se encuentra la información necesaria para poder descodificarlo.

Para codificar los datos en *base64* se hace uso de la clase Java "Base64.java" de Mikael Grev.

### 6.2.4 TOKENS DE CONTROL

Una vez se ha terminado de cifrar el mensaje, la última etapa del proceso consiste introducir en él los tokens de control que indiquen en el proceso del descifrado los tipos de cifrado que se han empleado para poder descifrarlo correctamente. Como se ha explicado ya en el diseño del sistema los tokens de control son los siguientes: @, #, $, & y %.

Estos tokens se concatenan al comienzo del mensaje una vez cifrado, desordenado y codificado para saber en qué posición están exactamente éstos. Es muy importante el orden en el que se concatenan estos tokens ya que será el orden en el que se han concatenado las claves de cada uno de estos métodos para generar la contraseña principal del mensaje.

El orden en el que se concatenarán estos tokens es: @, #, $, & y %. Esto quiere decir que, si un mensaje ha sido cifrado con todos los tipos de cifrado, comenzará de la siguiente forma: "%&$#@" (porque se concatena al principio del mensaje).

## 6.3 MÓDULO DE EXTRACCIÓN DEL MENSAJE INCLUIDO EN EL CÓDIGO QR

Antes de descifrar un mensaje incluido en un código QR es obligatorio obtenerlo. Esa tarea puede realizarse de 2 formas completamente diferentes: por medio de la cámara del dispositivo móvil o por medio de una imagen almacenada en la memoria del terminal.

Dependiendo de cuál de estos procedimientos se utilice, se utilizará una librería u otra.

### 6.3.1 CÁMARA

Una forma muy cómoda de obtener un mensaje incluido en un código QR es enfocarlo directamente con la cámara del smartphone. Para esto, es necesario que el smartphone posea una cámara integrada y que el mensaje se obtenga en tiempo real. La librería utilizada anteriormente se llama ZXing y es muy completa para descifrar y generar códigos QR pero, para poder utilizar su módulo de cámara, es necesario que su aplicación oficial llamada "Barcode Scanner" esté instalada en el dispositivo. Esta obligación se debe a que no existe otra forma de escanear por la cámara un código QR sin hacer uso directamente de la actividad principal de la aplicación oficial de ZXing. Por eso, es necesario utilizar otra librería para este procedimiento, ya que resulta muy tedioso obligar al usuario a instalarse otra aplicación.

#### *6.3.1.1 ZBAR*

La librería que se ha decidido utilizar para este fin se llama ZBar. El objetivo principal de esta librería es la lectura de códigos de todo tipo de códigos de barras (entre ellos, el código QR). Esta librería está desarrollada para varios sistemas operativos y entre ellos se incluye Android. Es una librería muy fácil de usar con actividades prediseñadas en las que se puede utilizar la cámara del dispositivo como lector QR. [21]

#### *6.3.1.2 leerCamara*

Para utilizar este método, se ha optado por modificar una clase de ejemplo prediseñada que incluía la librería. Esta clase pide permisos al sistema operativo para utilizar la cámara del dispositivo y busca en tiempo real códigos QR que se enfoquen con ésta. Si no encuentra ninguno, la clase seguirá buscando hasta que este procedimiento se cancele. En el caso de que se encuentre algún código QR, la clase intentará extraer el mensaje que contiene (cifrado o no) y lo enviará a la actividad principal de tratamiento del mensaje.

La forma en la que se ejecuta esta clase que usa la cámara es mediante la clase principal de tratamiento de mensaje. Cuando el usuario apriete el botón de leer por cámara, la aplicación ejecutará la clase principal de tratamiento de mensajes con la variable de cámara seleccionada. Cuando la clase detecta que esta variable está activada, llama a la clase de cámara mediante un Intent.

Un Intent es un objeto nativo de Android. Este objeto es una descripción abstracta de una operación que se va a realizar. En este caso, la operación no es otra que ejecutar

una clase como actividad y esperar la respuesta de ésta. Esta respuesta que espera la clase principal es el mensaje incluido en el código QR. [22]

Una vez se obtiene el mensaje, se comprobará si está cifrado o no. Si estuviese cifrado se procedería a ejecutar el módulo de descifrado cuya implementación se explicará más adelante. Si no fuese así, el mensaje se mostraría directamente.

### 6.3.2 FICHERO

A parte de la cámara, se ha decidido incluir un modo de lectura de códigos QR que, a pesar de que su implementación es sencilla, no existe en casi ninguna aplicación que se encuentre en el repositorio oficial de Android. Este procedimiento consiste en permitir que el usuario seleccione un archivo de imagen que contenga un código QR por medio del explorador de archivos de su terminal y la aplicación extraiga el mensaje incluido en él.

Para lograr este fin se ha decidido utilizar de nuevo la librería ZXing, ya que soporta este tipo de lectura de códigos QR sin necesidad de tener instalada su aplicación oficial y, además, se considera que es conveniente reutilizar librerías antes que importar nuevas cuando las librerías importadas anteriormente ya cuentan con estas funcionalidades.

Esta librería es capaz de leer los pixeles de una imagen, detectar si existe un código QR representado en ella por medio de reconocimiento de patrones y aislarlo de los demás elementos para, finalmente, extraer el mensaje que éste contiene.

#### 6.3.2.1 *pasarABlancoYNegro*

Una de las funcionalidades más interesantes de la aplicación consiste en la generación de códigos QR de distintos colores para identificar rápidamente cuando un código QR está cifrado (es un QRypt) y cuando no. A pesar del atractivo que suma al sistema, se produce un problema serio: cuando los colores no son muy distintos del blanco, el método principal de lectura del código QR no funciona.

Para que esto no ocurra, se ha considerado extremadamente necesario desarrollar un método aparte que se ejecute antes que transforme el código QR de la imagen a blanco y negro. Este método recibe únicamente el Bitmap con el código QR ya aislado de los demás elementos de la imagen gracias a la clase *MutiformatReader* de la librería ZXing.

| pasarABlancoYNegro(códigoQR):<br>• Se genera un Bitmap vacío que hará de salida con las mismas características que el que contiene el código.<br>• Se recorre pixel a pixel el Bitmap con el código:<br>  ○ Si el valor RGB del pixel se asemeja al blanco, se escribe un pixel blanco en el Bitmap de salida.<br>  ○ Si el valor RGB del pixel se asemeja a otro, se escribe un pixel negro en el Bitmap de salida.<br>• Se devuelve el Bitmap de salida. |
|---|

Tabla 122: Algoritmo del método *pasarABlancoYNegro*

Para comprobar si el color del pixel se asemeja al blanco, se descompone su color en valores RGB. Si el valor de cada uno de los valores RGB es superior a 220, el color que representa se considera blanco.

Se ha decidido interpretar los colores por medio de umbrales y no por los valores exactos de colores ya que es posible que la imagen que se está leyendo sea una foto y, por culpa de la luz, el color blanco es un gris muy claro. Si no se emplease la identificación del color por umbrales, ese color no sería blanco y lo interpretaría entonces como negro, quedando el código completamente ilegible.

#### 6.3.2.2 *leerFichero*

Al igual que en el procedimiento de la cámara, se hace uso de un objeto Intent para lograr el objetivo de este módulo pero, en vez de para obtener el mensaje del código que se está enfocando por la cámara, se utiliza para llamar al explorador de archivos del sistema operativo. Este explorador de archivos devolverá la ruta de la imagen que se ha seleccionado y la clase principal de tratamiento de mensajes interpretará los datos que ese encuentran en ella.

El método que se va a explicar a continuación es el principal en la lectura de códigos QR a partir de un fichero. Este método recibe por parámetros sólo la ruta de la imagen seleccionada.

leerFichero(ruta):

- Se transforma el archivo que existe en la ruta recibida a un objeto Bitmap.
- Se crean unos objetos RGBLuminanceSource y BinaryBitmap a partir del Bitmap que contiene toda la imagen en la que está incluida un código QR.
- Se aísla el código QR de los demás elementos de la imagen y se almacena en un Bitmap.
- Se pasa a blanco y negro el Bitmap con el código mediante el método *pasarABlancoYNegro*.
- Se extrae el mensaje del Bitmap con el código QR en blanco y negro utilizando el método *decode* de la clase *MultiFormatReader*.
- Se devuelve el mensaje extraído.

**Tabla 123: Algoritmo del método *leerFichero***

Al igual que en el procedimiento de lectura por cámara, una vez se obtiene el mensaje, se comprobará si está cifrado o no. Si estuviese cifrado se procedería a ejecutar el módulo de descifrado cuya implementación se explicará a continuación. Si no fuese así, el mensaje se mostraría directamente.

## 6.4 MÓDULO DE DESCIFRADO DE MENSAJES

Se ha explicado ya la implementación de los métodos principales del proceso de cifrado de mensajes en el que se codifica un texto de forma que su posterior lectura sin las contraseñas necesarias es imposible.

Como se explicó en el diseño del sistema, la aplicación hace uso de unos algoritmos de cifrado que se emplean en el mensaje para obtener una versión protegida de éste, pero también se explicó que utilizaría algoritmos de descifrado para poder obtener el mensaje original de nuevo. En este capítulo se van a explicar un poco más en profundidad estos últimos algoritmos.

Como se pudo ver en la Ilustración 16, el proceso de descifrado que se va a implementar empieza obteniendo los tokens y descodificando éste con *base64*, luego lo desordena y, finalmente, lo descifra generando la contraseña previamente.

### 6.4.1 TOKENS DE CONTROL

Lo primero que se realiza en este proceso es obtener los tokens de control para saber qué tipo de cifrado se le ha impuesto al mensaje y obtener así los datos principales para poder regenerar la contraseña de cifrado y descifrado.

El método que realiza esta tarea se llama *tokensDeControl* y recibe por parámetros únicamente el mensaje cifrado recién extraído del código QRypt.

tokensDeControl(mensajeCifrado):

- Se recorren las primeras posiciones del mensaje:
  - Si el caracter en la posición es un "%":
    - Se abre un diálogo y se pide al usuario que introduzca la contraseña.
    - Se genera un hash con esa contraseña con una función resumen SHA-1.
    - Se elimina el caracter de esa posición del mensaje.
  - Si el caracter en la posición es un "&":
    - Se abre un diálogo y se pide al usuario que introduzca el patrón.
    - Se genera un hash con ese patrón con una función resumen SHA-1.
    - Se elimina el caracter de esa posición del mensaje.
  - Si el caracter en la posición es un "$":
    - Se obtiene el Imei del terminal.
    - Se genera un hash con ese Imei con una función resumen SHA-1.
    - Se elimina el caracter de esa posición del mensaje
  - Si el caracter en la posición es un "#":
    - Se obtiene el modelo del terminal.
    - Se genera un hash con ese modelo con una función resumen SHA-1.
    - Se elimina el caracter de esa posición del mensaje
  - Si el caracter en la posición es un "@":
    - Se obtiene la fecha actual.
    - Se concatena el año, el mes y el día y se genera un hash con esa fecha con una función resumen SHA-1.
    - Se elimina el caracter de esa posición del mensaje
- Se concatenan los hashes.
- Se devuelven los hashes.

Tabla 124: Algoritmo del método *tokensDeControl*

Una vez se obtienen los hashes concatenados, se extraen los demás datos que se utilizarán para generar la clave. Estos datos se encuentran concatenados al final del mensaje y son: el *salt*, el número de iteraciones del proceso de derivación y el vector de inicialización del algoritmo de cifrado. Una vez extraídos del mensaje, se borran de este para no

### 6.4.2 DESCODIFICACIÓN

Como se ha implementado el sistema de codificación *base64*, hay que hacer uso de la descodificación *base64* para volver a obtener el mensaje cifrado desordenado en estructura de array de bytes, que será necesario para poder descifrarlo posteriormente. Además, en este proceso se extraen los demás datos que se utilizarán para generar la clave. Estos datos se encuentran concatenados al final del mensaje y son: el *salt* (su hash), el número de iteraciones del proceso de derivación y el vector de inicialización del algoritmo de cifrado.

### 6.4.3 REORDENACIÓN

En el módulo de cifrado se utilizó un algoritmo que desordenaba la posición de los datos en el mensaje. En esta tarea es menester devolver el mensaje desordenado a su

orden original para poder descifrarlo correctamente. El algoritmo utilizado, al igual que en la desordenación, recibe por parámetros un mensaje en formato array de bytes.

orden(mensaje):
- Se clona el mensaje en una variable auxiliar.
- Se declaran dos contadores: el punto de inicio del mensaje (posición 0) y el punto final (posición longitudMensaje-1).
- Se recorre todas las posiciones del auxiliar del mensaje salvo la última:
  - Se cambia el dato que está en la posición siguiente a la del contador de punto de inicio del mensaje por el valor de la posición que se está observando en el auxiliar.
  - Se incrementa en 1 este contador de posición inicial.
  - Se cambia el dato que está en la posición anterior a la del contador de punto final del mensaje por el siguiente valor de la posición que se está observando en el auxiliar.
  - Se resta el valor 1 al contador de la posición final.
  - se incrementa en uno el contador general del bucle.
- Si la longitud del mensaje es impar:
  - Se escribe el valor que se encuentra al final del auxiliar en la posición que indica el contador del punto final del mensaje.
- Se devuelve el mensaje ordenado.

Tabla 125. Algoritmo del método *orden*

Para un mejor entendimiento del algoritmo, se añade una ilustración a continuación con el código de este método.

```
public static byte[] orden(byte[] mensaje) {
    byte[] aux = mensaje.clone();
    int puntFin = mensaje.length - 1;
    int puntIni = 0;

    for (int i = 0; i < mensaje.length - 1; i++) {
        mensaje[puntIni++] = aux[i];
        mensaje[puntFin--] = aux[++i];
    }

    if (mensaje.length % 2 != 0)
        mensaje[puntFin] = aux[mensaje.length - 1];

    return mensaje;
}
```

Ilustración 42: Captura del método *orden*

Una vez se ejecuta este algoritmo, los datos que componen el mensaje quedarán reordenados a su posición original y sólo faltará descifrar el mensaje.

### 6.4.4 GENERACIÓN DE CONTRASEÑA Y DESCIFRADO DEL MENSAJE

La última tarea que queda realizar en este proceso es la de descifrado del mensaje. Este subproceso realizará cuasi los mismos pasos que el algoritmo de cifrado salvo que éste descifrará el mensaje.

Como se han obtenido los hashes de las contraseñas en la ejecución del algoritmo del método *tokensDeControl* y, también, se han conseguido el *salt*, el número de iteraciones y el vector de inicialización en la tarea de descodificación, sólo hace falta generar la clave y descifrar el mensaje con ella.

El método de descifrado recibirá por parámetros el mensaje ya ordenado pero cifrado y los hashes de las claves recogidas en el método de los tokens de control concatenadas (se le llamará contraseña a partir de ahora), el *salt*, el número de iteraciones y el vector de inicialización.

descifrar(mensaje, contraseña, *salt*, iteraciones, vector):

- Se concatena el *salt* con la contraseña.
- Se deriva la contraseña:
  - Se deriva la contraseña haciendo uso de la función de derivación "PBKDF2WithHmacSHA1" tantas veces como represente el número de iteraciones recogido en los parámetros.
- Se descifra el mensaje:
  - Se descifra el mensaje con la clave obtenida anteriormente y el vector de inicialización recibido por parámetros utilizando el algoritmo de cifrado AES.
- Se devuelve el mensaje descifrado.

**Tabla 126: Algoritmo del método *descifrar***

Es muy importante utilizar la misma función de derivación y el mismo algoritmo de cifrado (en este caso, de descifrado) para obtener el mensaje original. Cualquier cambio en los valores con los que se cifró el mensaje llevaría a generar un mensaje descifrado completamente distinto del original. Por eso es necesario conocer el *salt*, el número de iteraciones y el vector de inicialización previamente.

Una vez obtenido el mensaje, sólo haría falta mostrarlo para que el usuario lo conozca.

## 6.5 RESTO DE MÓDULOS

Una vez se han implementado los módulos importantes de la aplicación, queda implementar el conjunto de módulos que se han decidido realizar para completar el programa.

Estos módulos son los de cambio de ajustes del sistema y el de conocer las novedades de la versión del programa.

### 6.5.1 AJUSTES DEL SISTEMA

La función de este módulo es permitir al usuario modificar las preferencias básicas del sistema de información. Estas preferencias que podrá modificar serán: la ruta de la carpeta de salida donde se almacenarán los códigos QR y QRypt generados con el programa, y el idioma nativo de la aplicación.

#### *6.5.1.1 MEMORIA DEDICADA PARA LAS APLICACIONES*

Las aplicaciones del sistema operativo Android hacen una gran uso de las bases de datos SQlite. Almacenan en ellas toda la información de las preferencias del sistema e, incluso, hay algunas que almacenan datos del login del usuario (como, por ejemplo, "UC3Moid", que es un buen cliente de aula global pero tiene este fallo de seguridad). Son de rápido acceso pero tienen un grave problema: la falta de seguridad.

Estas bases de datos son muy rápidas pero no están cifradas. Se guardan en la ruta "/data/data/paqueteDeLaAplicacion/databases" que es una carpeta incluida en el sector "protegido" del sistema operativo. una aplicación, sin permisos de superusuario es incapaz de acceder a esta ruta pero, aun así, existen muchas aplicaciones en el repositorio de Android que son capaces de hacerlo. Cada vez son más los smartphones que tienen un kernel con permisos de superusuario y eso significa que la seguridad que ofrecía este sector "protegido" del sistema ya es nula.

Por eso, a lo largo del último año se ha vuelto una medida común el almacenar los datos importantes en la memoria dedicada a la aplicación. Su acceso es más rápido y es segura porque no puede acceder a ella ninguna otra aplicación que la propietaria; por lo tanto, eso la convierte en una buena medida de seguridad adicional.

De momento, los únicos datos que se guardarán en ella serán la ruta de la carpeta de salida y el idioma elegido para la aplicación pero, en versiones posteriores, se guardarán también las contraseñas favoritas y demás datos que necesitan estar almacenados en un lugar seguro.

La forma de acceder a esta memoria es muy simple: basta con hacer uso de la clase de Android *SharedPreferences* para obtener los datos como si fueran atributos: accediendo a estas preferencias por posiciones. De esta forma, en la posición 0 habrá un dato y en la posición 4 otro distinto y se podrán obtener desde cualquier actividad de la aplicación.

#### *6.5.1.2 MODIFICAR LA CARPETA DE SALIDA*

El método de modificación de la ruta de la carpeta de salida sólo recibe 1 atributo que es la cadena con la nueva ruta.

| |
|---|
| carpetaDeSalida(ruta):<br>• Se accede a la posición del objeto *SharedPreferences* que almacena la ruta de la carpeta de salida.<br>• Se sobrescribe su valor por el de la ruta que se recibe por parámetros. |

**Tabla 127: Algoritmo del método *carpetaDeSalida***

Cuando generar un código QR o QRypt, antes de almacenarse en la tarjeta SD del dispositivo, se accede a este objeto para obtener la ruta de la carpeta de salida y saber dónde generar el archivo.

La ruta por defecto, si el usuario no la cambia, es: "/sdcard/QRypt/". Por lo tanto, si no se modifica este valor, los códigos que se generen se guardarán ahí. Como los nombres de los archivos de imagen se generan con la fecha y hora de generación incluidos los segundos, siempre tendrán nombres diferentes y no se corre el riesgo de que se sobrescriban aunque no se cambie la ruta de esta carpeta.

#### 6.5.1.3 MODIFICAR EL IDIOMA DE LA APLICACIÓN

Hoy en día el hecho de desarrollar una aplicación que no esté traducida y localizada al inglés resulta verdaderamente extraño. La gran mayoría de las aplicaciones que se encuentran en el repositorio de Android están traducidas al inglés. Esto se debe a que es el idioma universal que, en los tiempos que corren, casi todo el mundo habla. El simple hecho de no traducir tu aplicación al inglés significa, probablemente, más de un 60% de menos descargas que si estuviese traducida. Por esta razón, se ha decidido traducir esta aplicación al inglés.

Aun así, en este apartado se ha querido ir un paso más adelante. La mayoría de las aplicaciones para Android que están traducidas no permiten cambiar el idioma de la aplicación directamente desde ella. Se muestran en el idioma que está configurado para todo el sistema operativo. De esta forma, si el sistema operativo está en castellano pero la aplicación está muy mal traducida y se prefiere en inglés, esto no se puede cambiar.

Para poder modificar el idioma de la aplicación en tiempo de ejecución y sin importar el idioma en el que esté el sistema operativo, se hace uso de dos clases: *Locale* y *Configuration*.

Cuando se realiza el diseño de una aplicación, todas las cadenas estáticas que se muestran en ella se guardan como variables finales para poder llamarlas sin tener que escribirlas directamente en el código. Estas variables se guardan en la carpeta "res/values/" (resources) y, dependiendo del idioma que sea, en una subcarpeta o en otra. De esta forma, si el idioma de las cadenas escritas es inglés estadounidense, la subcarpeta dentro de "res" se llamará "values-en_US".

Como esta aplicación tiene que poder mostrarse en inglés y en español, cuenta con 2 subcarpetas dentro de res con 2 archivos "strings.xml" con las cadenas estáticas de un idioma y otro. El archivo "string.xml" con las cadenas en español está en "res/values-es_ES/" y el archivo "string.xml" en inglés está en "res/values-es_US". [23]

Por defecto, se elige el idioma que se tenga configurado en el dispositivo pero, si ese idioma no es ninguno de estos 2, la aplicación se mostrará en inglés. Si, por otro lado, el usuario decide cambiar el idioma, la aplicación modificará el valor del objeto *SharedPreferences* que representa el idioma al lenguaje que éste elija. Como se guarda en esta memoria dedicada a la aplicación, a partir de ese momento, el sistema se mostrará en ese idioma al no ser que se desinstale la aplicación o se vuelva a modificar éste.

Al iniciarse la aplicación, se ejecuta una función que comprueba este valor en la memoria y actualiza todas las cadenas que se muestren en la interfaz al idioma que allí figure.

establecerIdioma():
- Acceder a la posición de *SharedPreferences* donde se encuentra el valor del idioma.
- Crear un objeto *Locale* e inicializarlo con el idioma recogido.
- Seleccionar ese idioma como por defecto con el método *setDefault* de la clase *Locale*.
- Crear un objeto *Configuration*.
- Establecer ese idioma en la configuración con el atributo *locale* de la clase *Configuration*.
- Actualizar la interfaz con el método *updateConfiguration*.

**Tabla 128: Algoritmo del método *establecerIdioma***

En la tabla anterior se muestra el algoritmo del método para establecer el idioma que se ejecuta cada vez que se abre la aplicación.

Otro método importante que conviene explicar, aunque sea muy simple, es el de modificar el idioma. Cada opción que se muestra en el diálogo que aparece en la interfaz de selección de idioma tiene vinculado una variable que almacena el idioma que se representa. De esta forma, cuando se pulsa el botón "English" se actualiza una variable al valor "en_US" y, cuando se selecciona el "Español", la variable toma el valor "es_ES". Por eso, cuando el usuario pulsa sobre la opción de modificar el idioma y selecciona uno de ellos y pulsa el botón aceptar, la aplicación envía el valor de esta variable al objeto *SharedPreferences* y lo almacena en la posición que represente el idioma. En la siguiente tabla se muestra el algoritmo que se utiliza para lograr tal fin.

actualizarIdioma():
- El usuario selecciona una de las opciones de idioma.
- Si el usuario pulsa "Aceptar":
  - Se envía el valor del idioma escogido a *SharedPreferences* y se almacena en la posición correspondiente.
  - Se ejecuta la función establecerIdioma(). (Explicada anteriormente)
- Si el usuario pulsa "Cancelar":
  - Se cierra el diálogo y no se hace nada.

**Tabla 129: Algoritmo del método *actualizarIdioma***

### 6.5.2 MOSTRAR NOVEDADES DE LA VERSIÓN

Cuando se actualiza una aplicación tanto desde el mismo programa como desde el repositorio oficial de Android, el sistema ya no es el mismo que antes. Los cambios

que ofrece una versión de un sistema de información pueden ser desde mínimos (como arreglos de velocidad) hasta inmensos (como uno cambio radical en el diseño, o una implementación de una función nueva). Sea el cambio que sea, el usuario debe saber qué es lo que le ofrece la nueva versión que no le ofrecía la anterior para valorar si conviene actualizar la aplicación o si, por el contrario, no es algo importante y puede esperar a otra nueva.

El repositorio oficial de Android ("Google Play", otrora "Market") ofrece esta funcionalidad pero la gran mayoría de la gente actualiza automáticamente las aplicaciones y no puede leer estas novedades. Si se ha cambiado algo puede resultar chocante al usuario y, por obligación, va a tener que abrir la aplicación "Google Play" para enterarse de qué novedades ha habido pero, implementando un pequeño log que almacene las novedades y que se muestre si el usuario quiere, éste no tendría que salir de la aplicación para enterarse de esta información.

El algoritmo que expone la función que se encarga de satisfacer esta necesidad se muestra a continuación:

| mostrarLog():<br>• Buscar la variable en el archivo "strings.xml" del idioma elegido que contenga la información.<br>• Mostrar esta cadena por pantalla en un diálogo. |
|---|

**Tabla 130: Algoritmo del método *mostrarLog***

# 7 EVALUACIÓN Y RESULTADOS

En esta sección se pondrá a prueba la aplicación desarrollada y se mostrarán los resultados obtenidos para, finalmente, comprobar su funcionalidad y concluir con una sentencia sobre ella.

## 7.1 FUNCIONALIDAD

En este primer capítulo se va a comprobar si la aplicación realiza todas las funcionalidades que se propusieron en los requisitos. Sólo se evaluará si realiza todas las funcionalidades y si lo hace bien, en un capítulo posterior se evaluará su rendimiento.

### 7.1.1 GENERACIÓN DE CÓDIGOS QR

La primera funcionalidad que se va a evaluar es la de generación de códigos QR simples. El requisito de usuario que exigía esta funcionalidad es el RUC-01.

Cuando se pulsa sobre el botón de generación de códigos QR en la interfaz principal, la aplicación abre una nueva actividad que permite introducir un mensaje con un máximo de 2000 caracteres. Una vez se introduce uno y se pulsa en el botón de generar el código QR, la aplicación abre otra actividad y muestra el código QR generado.

Por ejemplo, si intentamos generar un código QR con el mensaje "Esto es una prueba":


Ilustración 43: Prueba de generación de QR 1

Se observa en la ilustración anterior que la aplicación permite escribir un mensaje y que, además, lleva la cuenta en tiempo real de los caracteres escritos hasta el momento.


Ilustración 44: Prueba de generación de QR 2

En la ilustración anterior se muestra como la aplicación ha generado un código QR a partir de un mensaje y en la parte superior se encuentra la opción para compartirlo a otra aplicación. A continuación, se comprobará por medio de otro programa si el código QR generado contiene el mensaje a partir del cual se ha generado. A pesar de que esta aplicación puede leer códigos QR, se utiliza otra aplicación para esta prueba para comprobar si el código QR generado es universal.

Para esta prueba se utilizará la aplicación lectora de códigos de barra por antonomasia: "Barcode Scanner".


Ilustración 45: Prueba de generación de QR 3

Como se observa en la ilustración anterior, la aplicación "Barcode Scanner" ha leído en código QR el mensaje "Esto es una prueba" que es, precisamente, el mensaje que se ha escrito anteriormente. Para eliminar dudas, se ha decidido realizar esta prueba leyendo este código directamente desde la imagen ya insertada en la memoria (como se puede ver en la captura del móvil).

Hasta este punto se ha evaluado la funcionalidad de generación de códigos QR, la siguiente que se evaluará es la de compartición de esto código con una aplicación directamente desde el menú superior de la pantalla de muestra del código generado.

### 7.1.2 COMPARTICIÓN DEL CÓDIGO CON APLICACIONES DE TERCEROS

Cuando se genera un código QR (o QRypt), la aplicación ofrece al usuario la funcionalidad de compartir éste con otra aplicación para enviarlo a un contacto o de publicarlo en Internet. Esta funcionalidad viene exigida por el requisito de usuario RUC-14.

Una vez se ha generado el código QR y se muestra en la interfaz diseñada para la muestra del código, aparece en la parte superior un botón de compartición del código. Si se pulsa éste, se extiende un menú en el que se puede elegir la aplicación a la que se puede enviar el código.

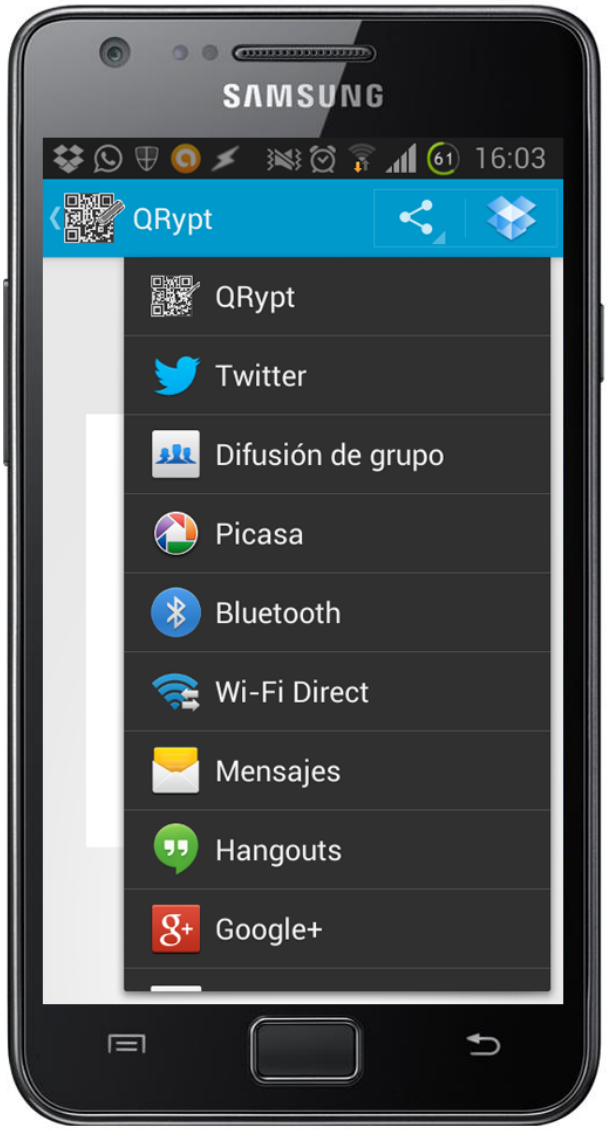



Ilustración 46: Prueba de compartición 1

En la ilustración anterior se observa cómo, al pulsar en el botón de compartición, se extiende un menú para seleccionar la aplicación a la que se quiere enviar el código.

A continuación, se van a mostrar algunos ejemplos de cómo funciona el sistema si se intenta enviar el código a otra aplicación.

#### 7.1.2.1 GMAIL

En el diseño del sistema se definió que, cuando se eligiese la opción de enviar un código por GMail, lo que la aplicación abriría sería el módulo de redacción de un mensaje del programa oficial de GMail y adjuntaría automáticamente la imagen con el código QR generado.


Ilustración 47: Prueba de compartición 2 (GMail)

Como se observa en la ilustración anterior, se abre el módulo descrito anteriormente de la aplicación GMail y se adjunta automáticamente el código QR a éste. Sólo habría que escribir un remitente, un asunto y un mensaje ya que la tarea de adjuntar la imagen ya se ha cubierto mediante la aplicación desarrollada en este proyecto.

#### 7.1.2.2 DROPBOX

En el diseño del sistema se definió que, cuando se eligiese la opción de enviar un código a Dropbox, lo que la aplicación abriría sería el módulo de subida de archivo del programa oficial de Dropbox.

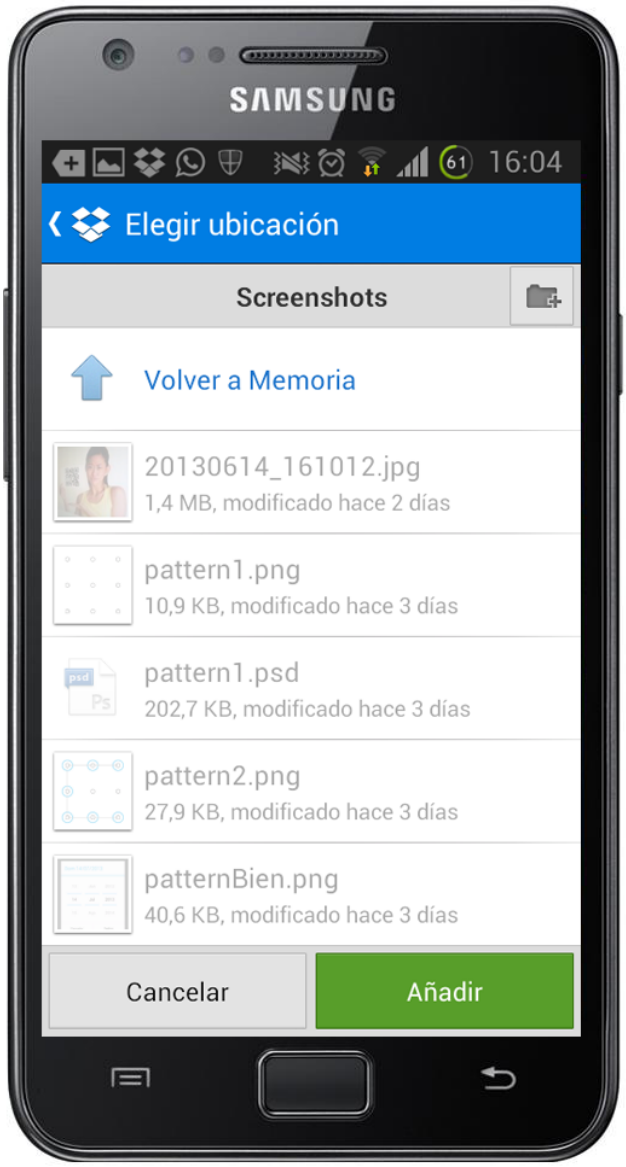



Ilustración 48: Prueba de compartición 3 (Dropbox)

Como se observa en la ilustración anterior, se abre el módulo descrito anteriormente de la aplicación Dropbox y se da a elegir al usuario la carpeta en la que quiere almacenar la imagen con el código QR. Sólo habría que entrar en la carpeta y pulsar el botón "Añadir" ya que la tarea de elegir el archivo que subir ya se ha cubierto mediante la aplicación desarrollada en este proyecto.

#### *7.1.2.3 TWITTER*

En el diseño del sistema se definió que, cuando se eligiese la opción de enviar un código a Twitter, lo que la aplicación abriría sería el módulo de redacción de *tweet* de del programa oficial de Twitter y adjuntaría en él la imagen con el código.


Ilustración 49: Prueba de compartición 4 (Twitter)

Como se observa en la ilustración anterior, se abre el módulo descrito anteriormente de la aplicación Twitter. Sólo habría que escribir el *tweet* y pulsar sobre el botón "Tweet" ya que la tarea de elegir el archivo que adjuntar ya se ha cubierto mediante la aplicación desarrollada en este proyecto.

#### *7.1.2.4 WHATSAPP*

En el diseño del sistema se definió que, cuando se eligiese la opción de enviar un código a Whatsapp, lo que la aplicación abriría sería el módulo de compartición de un imagen con un contacto.

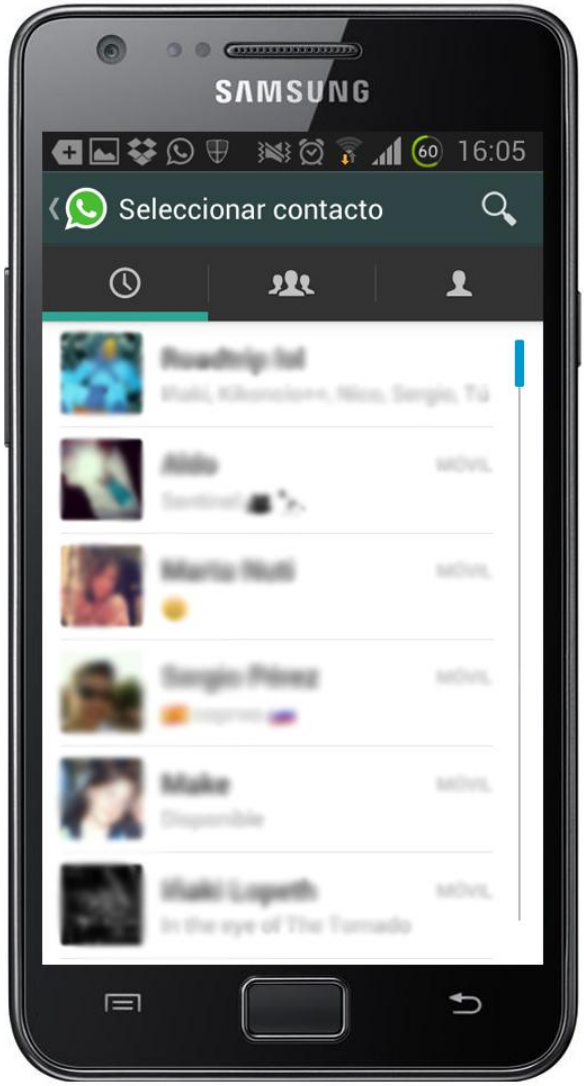



Ilustración 50: Prueba de compartición 5 (Whatsapp)

Como se observa en la ilustración anterior, se abre el módulo descrito anteriormente de la aplicación Whatsapp. Sólo habría que elegir el contacto al que enviar la imagen como se muestra en la ilustración siguiente:


Ilustración 51: Prueba de compartición 6 (Whatsapp)

El programa pide confirmación del usuario para enviar la imagen. Si el usuario pulsa el botón "Ok", la imagen se enviará; si, por otro lado, pulsa el botón "Cancelar", el programa volverá a la lista de contactos.

La siguiente imagen muestra cómo se envía el código cuando se pulsa sobre el botón "Ok":


Ilustración 52: Prueba de compartición 7 (Whatsapp)

Como se ha podido observar, el código se envía correctamente al contacto elegido.

En este capítulo se han realizado las pruebas del módulo de compartición y se ha mostrado cómo la aplicación realiza esta funcionalidad tal y como estaba descrita.

### 7.1.3 GENERACIÓN DE CÓDIGO QRYPT

Otra funcionalidad muy importante que la aplicación debía implementar era la de poder generar un código QR con un mensaje cifrado; o sea, un código QRypt. El requisito de usuario que exige esta funcionalidad es el RUC-02.

En esta sección se va a comprobar que el programa ofrece la posibilidad de generar un código QRypt y que, además, lo genera. La comprobación de si el código QRypt contiene el mensaje "Esto es una prueba", se realizará en una sección posterior en la que se demuestre que la aplicación puede leer los códigos cifrados por ella.

Como se describió en el diseño del sistema, este módulo es idéntico al de generación de un código QR pero con algunas opciones más que ofrece el programa antes de generar el código.


Ilustración 53: Prueba de generación de QRypt 1

En la ilustración anterior se muestra cómo el programa vuelve a permitir al usuario introducir un mensaje pero, esta vez, el mensaje se cifrará antes de ser incluido en un código.

Si se elige la opción, por ejemplo, de generar el código de color rosa y cifrar el mensaje con el imei del teléfono, se muestra el código en la misma interfaz que el anterior y con las mismas opciones de compartición.


Ilustración 54: Prueba de generación de QRypt 2

Como se puede observar, el código generado, además de ser del color elegido es notablemente más grande que el código QR anteriormente mostrado. Esto se debe a que si se recuerda de secciones anteriores, este QRypt no sólo contiene el mensaje, si no, además, el salt, el número de iteraciones y el vector de inicialización. Además, si se compara la ilustración anterior con la siguiente, se podrá comprobar cómo, debido al vector de inicialización, los 2 códigos QRypt generados son distintos. Cabe añadir que ambos contienen el mensaje "Esto es una prueba" y están cifrado con el imei del teléfono.


Ilustración 55: Prueba de generación de QRypt 3

Por último, en la siguiente ilustración se muestra cómo, utilizando el programa "Barcode Scanner", el mensaje contenido en estos QRypt que se obtiene es completamente ilegible.


Ilustración 56: Prueba de generación de QRypt 4

Se puede observar en la ilustración anterior cómo el mensaje comienza con un token de control, precisamente el que corresponde con el cifrado por imei.

### 7.1.4 LECTURA DE CÓDIGO POR CÁMARA

Una vez evaluadas las funcionalidades relacionadas con la generación de códigos, es el turno de valorar los resultados obtenidos con las pruebas realizadas relacionadas con la lectura de códigos. En este capítulo se comprobará la funcionalidad de lectura de códigos utilizando la cámara del dispositivo. El requisito de usuario que exige esta funcionalidad es el RUC-03.

#### *7.1.4.1 CÓDIGO QR*

Se va a comprobar con el código generado en el capítulo de evaluación del módulo de generación de códigos QR si la aplicación es capaz de leer el código y recuperar el mensaje al igual que la aplicación "Barcode Scanner".

Cuando el usuario pulsa el botón de lectura por cámara, el programa muestra la interfaz de la cámara. Desde ésta, el usuario enfoca el código QR y la clase de la cámara obtiene el mensaje y lo muestra por pantalla.

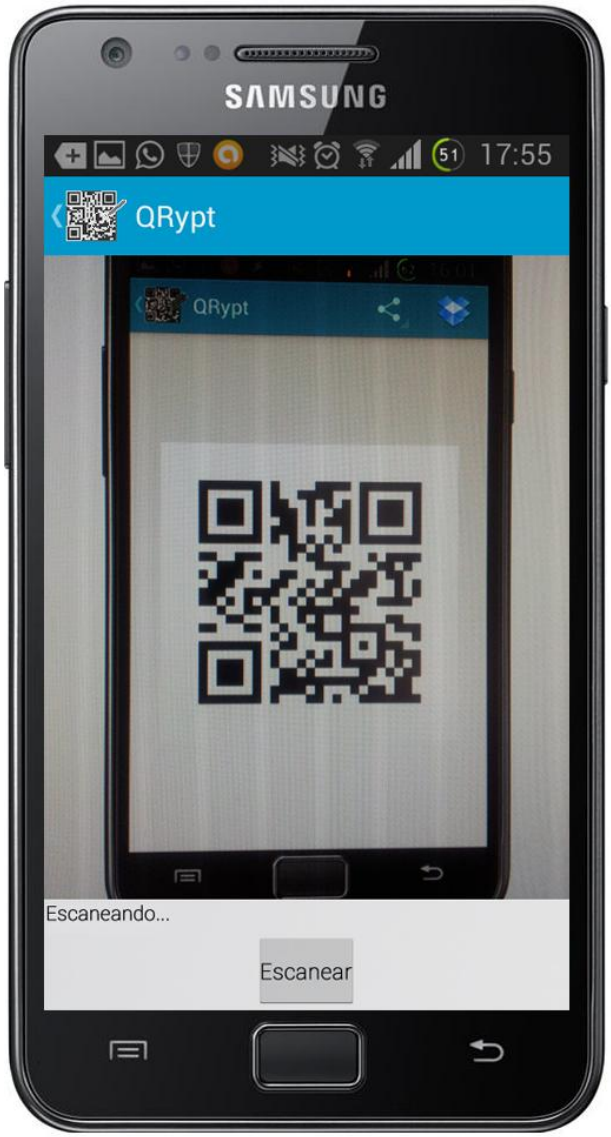



Ilustración 57: Prueba de lectura por cámara 1

En la ilustración anterior se muestra cómo se ejecuta la interfaz de la cámara y permite que el usuario enfoque con ella el código que quiere leer. Una vez se ha enfocado, la aplicación procederá a extraer el mensaje.


Ilustración 58: Prueba de lectura por cámara 2

Como el mensaje contenido en el código no estaba cifrado, la aplicación lo muestra directamente por pantalla. Se observa en la ilustración anterior como la aplicación ha extraído con éxito el mensaje.

#### 7.1.4.2 CÓDIGO QRYPT

En el caso del QRypt, una vez la aplicación obtenga el mensaje, verá que éste está cifrado y procederá a descifrarlo según el tipo de cifrado que se le haya impuesto. Se van a realizar un par de pruebas con códigos QRypt cifrados de distinta forma para observar la interactuación del programa con el usuario.

Si se intenta leer uno de los códigos QRypt generados en la sección de evaluación del módulo de generación QRypt:

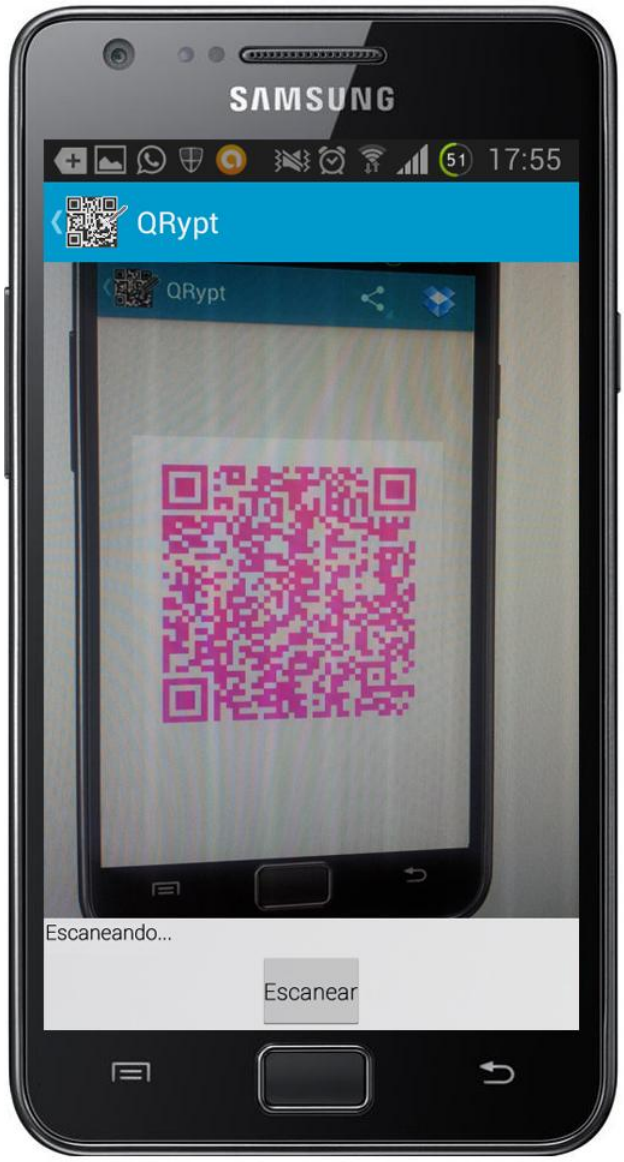



Ilustración 59: Prueba de lectura por cámara 3

La aplicación extraería el mensaje y leería el token de control "$". Al leerlo, se identifica que está cifrado con el imei del dispositivo e intenta obtener el imei del terminal que está leyendo el código. Una vez consigue el imei, la aplicación intenta descifrar el mensaje y lo muestra.


Ilustración 60: Prueba de lectura por cámara 4

Si el terminal lector es el mismo que el generador (tienen el miso imei), el mensaje se muestra correctamente como se observa en la imagen anterior.

Si, por el contrario, el dispositivo lector no es el mismo que el generador, al estar cifrado el mensaje con el imei del segundo, el campo del mensaje se muestra vacío ya que, al intentar descifrar el mensaje con una clave errónea, ocurre una excepción.

Si, se genera un código QRypt que cifre el mensaje con alguno de los tipos de cifrado que exigen una interacción del usuario, al leer el código se pediría al usuario que introduzca algún dato.

Por ejemplo, si se genera un código QRypt con el mensaje "Esto es una prueba 2" cifrado con una contraseña, la aplicación pide al usuario que introduzca esta contraseña como se muestra en la siguiente ilustración:


Ilustración 61: Prueba de lectura por cámara 5

Cuando se introduce esta contraseña, el código se termina de generar y se muestra por pantalla como se ha explicado antes.


Ilustración 62: Prueba de lectura por cámara 6

Cuando se intenta leer este código mediante la cámara del dispositivo, la aplicación detecta que está cifrado con una contraseña y pide al usuario que la introduzca.


Ilustración 63: Prueba de lectura por cámara 7

Una vez se introduce la contraseña, la aplicación intenta descifrar el código con ella. Como la contraseña para cifrar y descifrar era exactamente la misma, en la siguiente ilustración se muestra como el mensaje ha sido descifrado correctamente.


Ilustración 64: Prueba de lectura por cámara 8

### 7.1.5 LECTURA DE CÓDIGO DESDE UN FICHERO

la otra forma de leer un código QR o QRypt es eligiendo desde el explorador de archivos una imagen que represente uno. El requisito de usuario que exige esta funcionalidad es el RUC-04.

Como ya se ha mostrado en el capítulo anterior mediante captura cómo funciona el proceso de lectura de un código e, incluso, el proceso de descifrado, este capítulo sólo se va a centrar en la única parte que cambia con respecto a la lectura por cámara: la forma de obtener el mensaje.

#### *7.1.5.1 ELEGIR LA IMAGEN DESDE LA APLICACIÓN*

La primera forma de elegir el código que se quiere leer es pulsando sobre el botón de lectura desde fichero. De esta forma, la aplicación abre el explorador de archivos al no ser que exista más de uno, en ese caso, muestra un menú para elegir el que realizará la tarea.

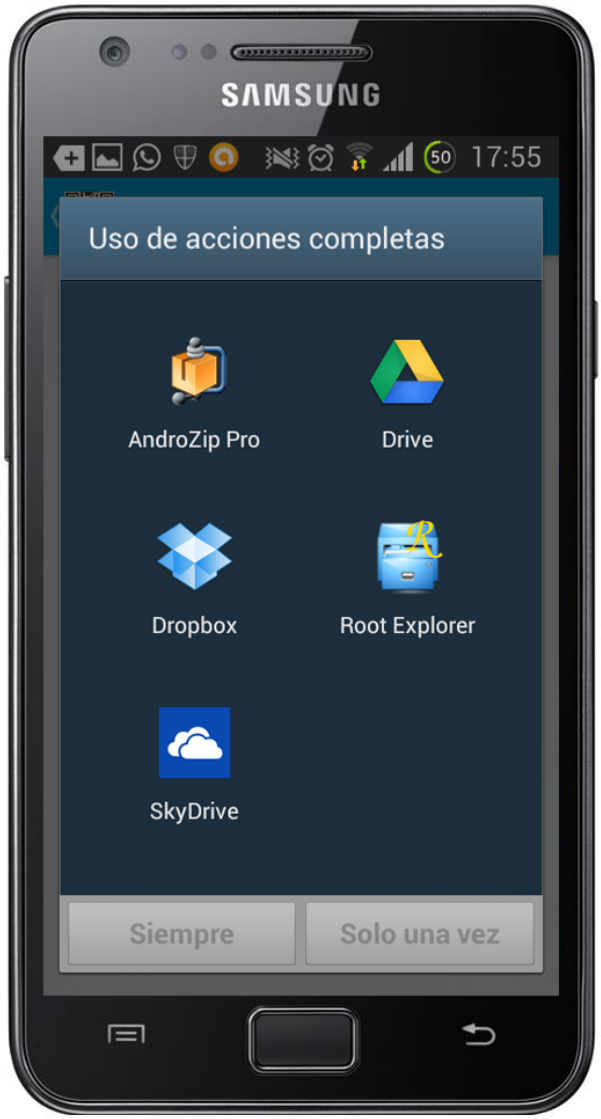



Ilustración 65: Prueba de lectura por fichero 1

Cuando se elige el explorador que se quiere usar, sólo basta con elegir el archivo desde él y el programa intentará obtener el mensaje incluido en el código QR o QRypt.


Ilustración 66: Prueba de lectura por fichero 2

En la ilustración anterior se muestra lo que mostraría si se eligiese la aplicación "Root Explorer" para escoger el archivo de imagen que se quiere leer.


Ilustración 67: Prueba de lectura por fichero 3

Como se ha elegido el archivo que se generó en la evaluación del módulo de generación QR, el mensaje que se muestra en la ilustración anterior es el mismo. Aun así, se ha comprobado que es posible elegir un código QR o QRypt que está incluido en una imagen desde un explorador de archivos y obtener el mensaje que éste guarda.

#### 7.1.5.2 ELEGIR LA IMAGEN DESDE LA GALERÍA

Aunque ya se ha demostrado que funciona la obtención del mensaje de un código por medio de la selección de una imagen desde el explorador de archivos, este explorador se abre desde la aplicación desarrollada. Se ha implementado un módulo de lectura de imágenes que permite seleccionar una imagen desde cualquier explorador e, incluso, aplicación que pueda manejar imágenes para enviarlas, directamente, a la aplicación desarrollada y que ésta las trate en busca de códigos.

Un ejemplo de esto que, además, sirve para evaluar esta funcionalidad es el siguiente:

Se abre una aplicación cualquiera que permita manejar fotografías y se busca una imagen con un código QR o QRypt en ella. Por ejemplo, "QuickPic", que es una galería.

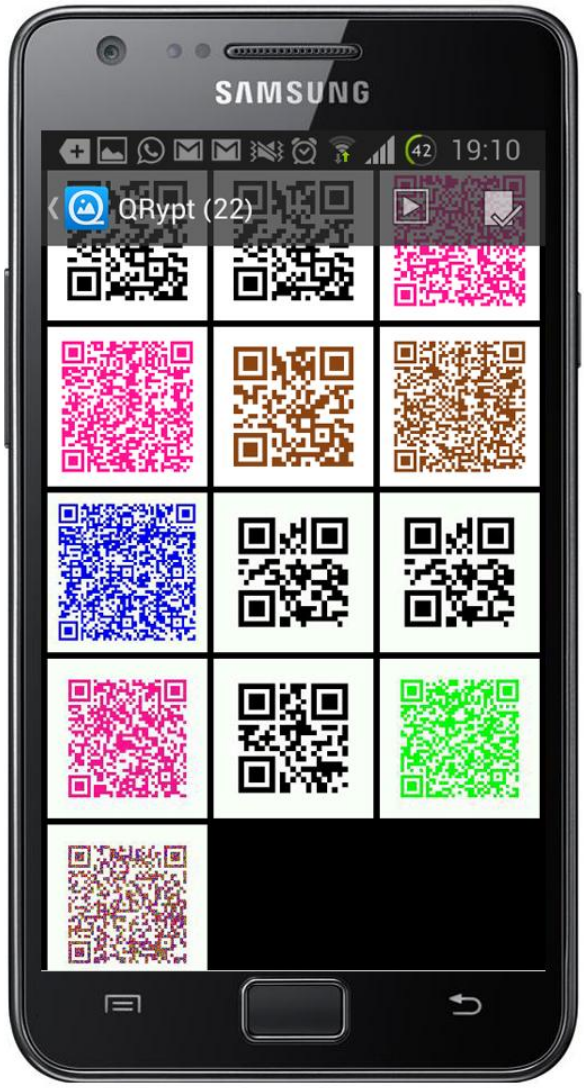



Ilustración 68: Prueba de lectura por fichero 4

Una vez se elige la imagen que se quiere leer, se busca en el menú de la aplicación el botón "Compartir" para enviarla a la aplicación desarrollada de la misma forma que ella se lo manda a otras aplicaciones.

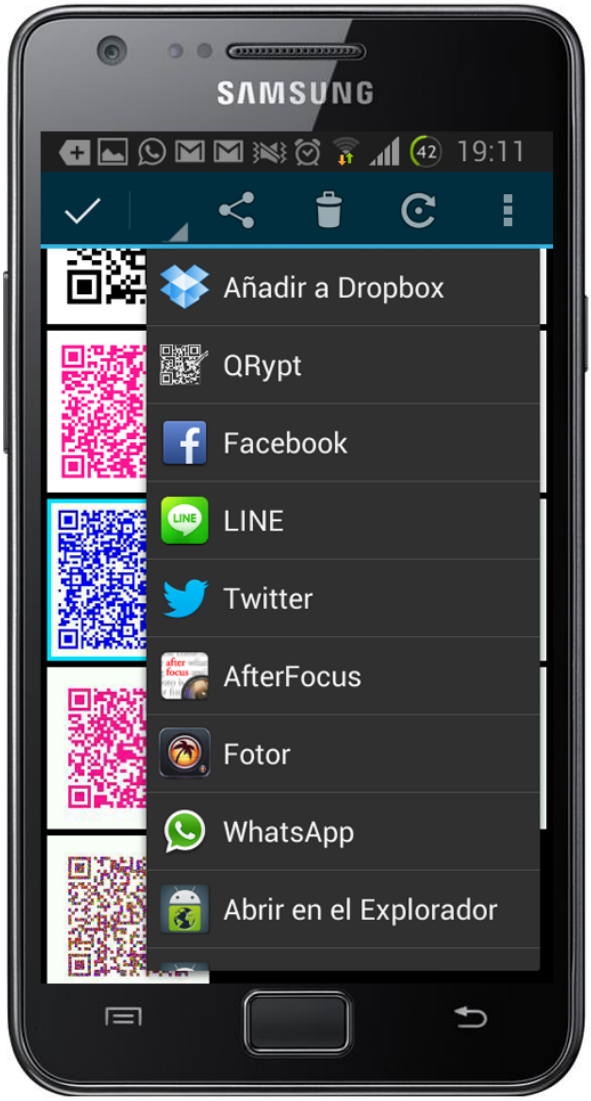



Ilustración 69: Prueba de lectura por fichero 5

Como se observa en la ilustración anterior, se va a intentar leer la imagen con el código QRypt azul. Cuando se mantiene pulsada, aparece un menú en el que se puede elegir la opción de compartir y, además la aplicación que se quiera. En el menú aparece la aplicación desarrollada ("QRypt"). Una vez se pulse en ella, ésta leerá la imagen e intentará descifrar el mensaje que se encuentra en su interior.


Ilustración 70: Prueba de lectura por fichero 6

Como el mensaje del código estaba cifrado con el modelo del terminal, no ha habido ningún problema a la hora de descifrarlo. Si estuviese cifrado con una contraseña o un patrón, la aplicación exigiría al usuario que lo introdujese.

Se ha observado que esta pequeña expansión del módulo de lectura por fichero también ha funcionado. Sólo queda evaluar la funcionalidad del módulo de cambio de ajustes y de muestra de novedades de la versión.

### 7.1.6 CAMBIO DE AJUSTES

EL módulo de cambio de ajustes cuenta con 2 funcionalidades: cambiar la ruta de la carpeta de salida y modificar el idioma en que se muestran las cadenas de caracteres de la aplicación.

#### 7.1.6.1 CARPETA DE SALIDA

Como se explicó en el diseño del sistema, la aplicación tenía que ser capaz de permitir que el usuario cambie la carpeta de salida donde se guardan los códigos generados desde la aplicación. El requisito de usuario que exige esta funcionalidad es el RUC-15.

Por defecto, la ruta en la que se guardan los códigos generados es "sdcard/QRypt". En la siguiente imagen se puede ver cómo todos los códigos generados se guardan allí:

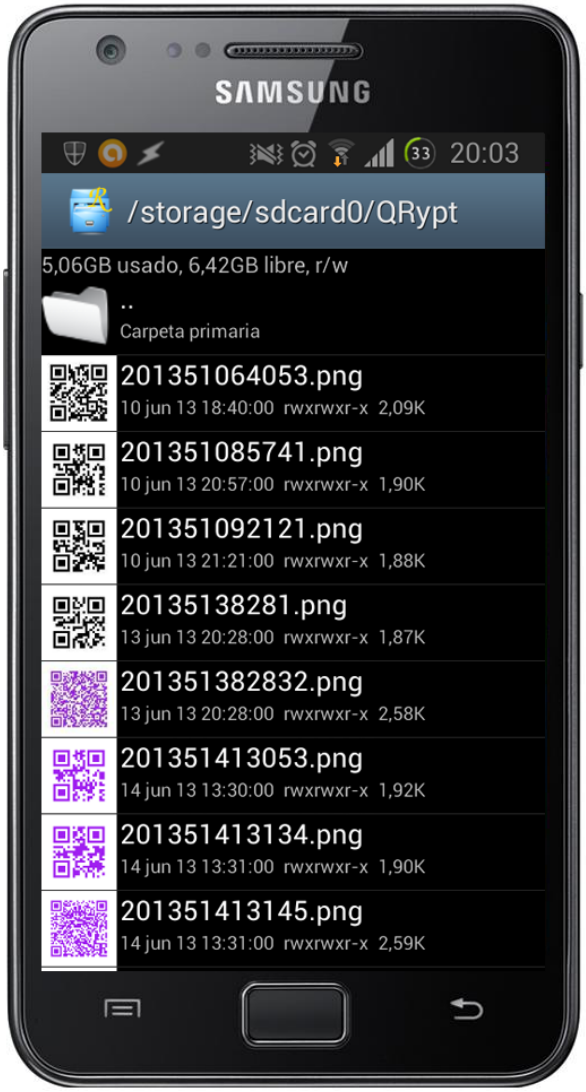



Ilustración 71: Prueba de carpeta de salida 1

Como dependiendo de la versión de Android y del terminal, el nombre la carpeta raíz de la tarjeta SD cambia, la aplicación comprueba cuál es este nombre y genera la ruta con él. De esta forma, la ruta por defecto donde se guardan los códigos generados en el terminal de la ilustración anterior no es "sdcard/QRypt", sino "sdcard0/QRypt".

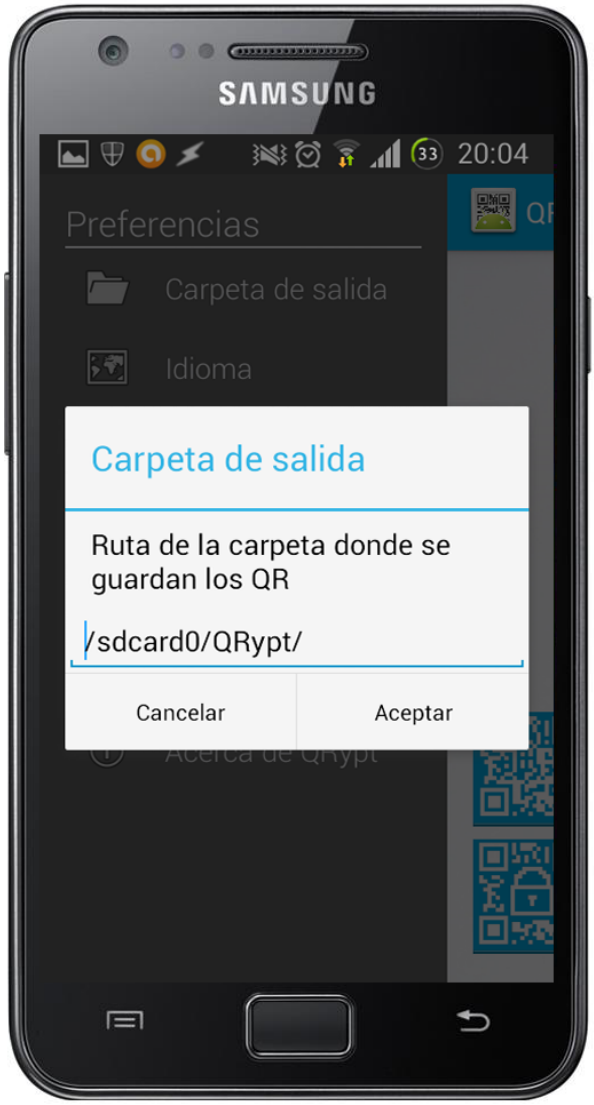



Ilustración 72: Prueba de carpeta de salida 2

En la ilustración anterior, se observa cómo la ruta en la que se guardan los códigos es la descrita en el párrafo anterior. Si esta ruta se modificase, los demás códigos que se generen después del cambio se guardarán en la nueva ruta.

Para comprobar que esto funciona, en esta prueba se modifica la ruta por "sdcard0/codigosQR/" como se muestra en la siguiente ilustración:

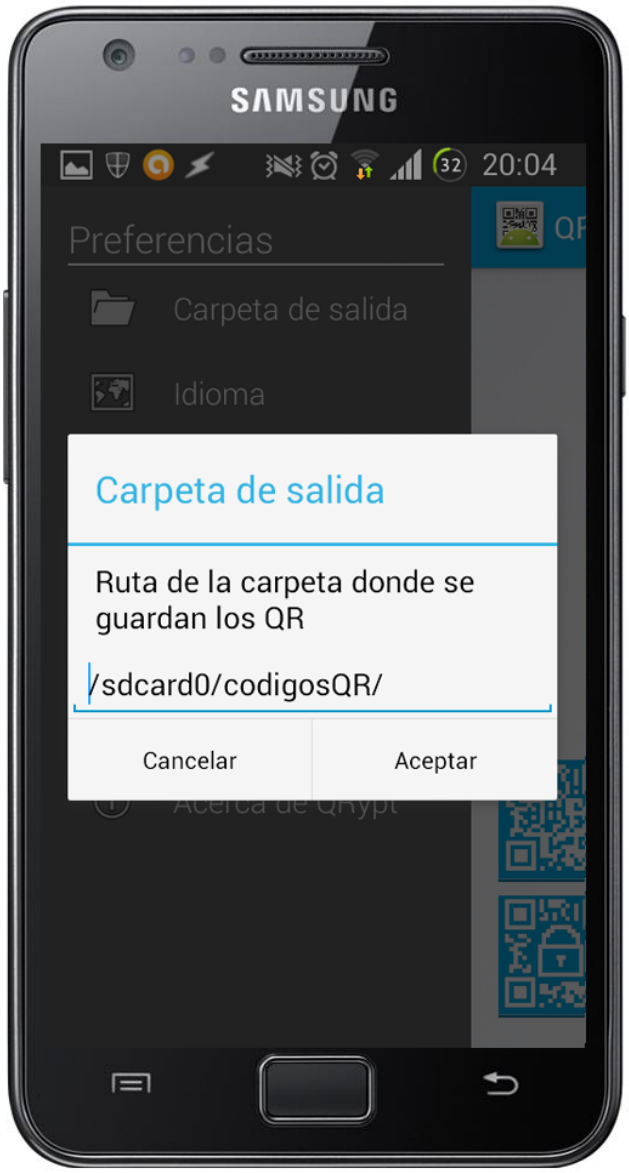



Ilustración 73: Prueba de carpeta de salida 3

Ahora que se ha modificado la ruta, los códigos que se generen deberían almacenarse en la nueva ruta. En la siguiente imagen se observa cómo, después de generar otro código, éste se ha guardado en la carpeta que se encuentra en la ruta especificada:

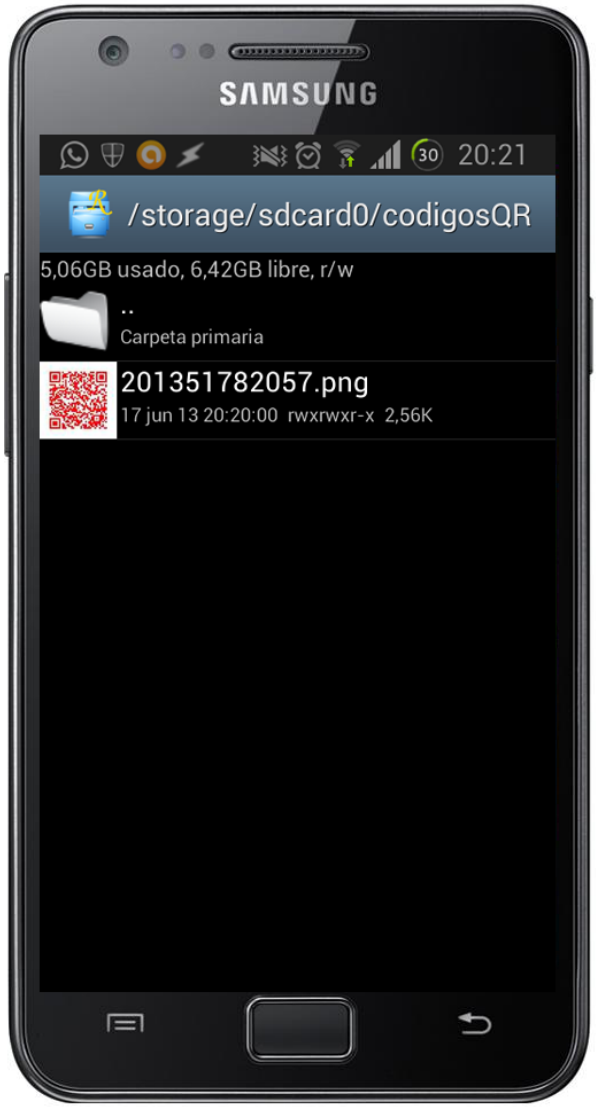



Ilustración 74: Prueba de carpeta de salida 4

Después de esta simple prueba, se puede decir que esta funcionalidad también obtiene su objetivo de forma óptima.

#### 7.1.6.2 CAMBIO DE IDIOMA

Como se ha explicado antes, modificar el idioma de la aplicación es una opción que ya ofrece el sistema operativo Android nativamente pero sólo funciona cuando se modifica el lenguaje de todo el sistema operativo.

Esta función ofrece al usuario la posibilidad de cambiar el idioma de la aplicación sin influir en el sistema operativo ni en las demás aplicaciones instaladas en el terminal.

La prueba que se va a realizar es muy simple: se accederá al panel para modificar esta opción del español al inglés y se verán los cambios producidos.


Ilustración 75: Prueba de cambio de idioma 1

Como se observa en la ilustración anterior, el diálogo permite elegir el idioma en el que se quiere que se muestre la aplicación. Al seleccionar el español, la aplicación queda como en todas las demás ilustraciones que se han mostrado hasta ahora pero, al seleccionar el inglés, la aplicación cambia. En la siguiente ilustración se muestra la pantalla principal cuando se ha pulsado el botón "Aceptar" del menú que aparece en la imagen anterior.


Ilustración 76: Prueba de cambio de idioma 2

Se puede observar en la última imagen cómo la interfaz de la aplicación ha cambiado y se ha traducido todo al inglés, lo que lleva a pensar que este módulo también ha desempeñado su función de forma correcta.

### 7.1.7 MOSTRAR LAS NOVEDADES DE LA VERSIÓN

Por último, queda evaluar si el último de los módulos que se iba a implementar funciona correctamente. La funcionalidad consiste en mostrar al usuario el número de versión y las novedades que ella implementa. Esta funcionalidad viene exigida por el requisito de usuario RUC-17.

La prueba es muy simple: consiste en pulsar el botón del panel deslizable dedicado a mostrar las novedades. Si se muestra el diálogo correspondiente, el resultado será el esperado.


Ilustración 77: Prueba de novedades de la versión

Como se observa en la imagen anterior, el diálogo que se muestra es el correcto.

### 7.1.8 CONCLUSIÓN

Se han puesto a prueba todas las funcionalidades que ofrece el programa y éste ha resuelto los problemas con los resultados que se esperaban. Cuando se le pide que genere un código QR, éste lo hace. Cuando se le pide que genere un QRypt, éste pregunta el color y el tipo de cifrado, en vez de generarlo directamente. Cuando se le pide que lea un código por cámara o por fichero, la aplicación lo hace correctamente. Además, el sistema también es capaz de cambiar el idioma, la ruta de la carpeta de salida y de mostrar las novedades de la versión que se han implementado.

Se recuerda que el dispositivo móvil utilizado para las pruebas posee un procesador dual-core de 1.2GHz.

## 7.2 RENDIMIENTO

Después de evaluar la funcionalidad de la aplicación, la siguiente característica que se va a poner a prueba es el rendimiento global del sistema. Este rendimiento vendrá dado por la velocidad con la que es capaz de realizar las funciones que precisen de más cómputo. Estas funciones son: generar códigos QR y QRypt. Las funciones de lectura de códigos no van a evaluarse ya que, una vez se ha enfocado el código con la cámara o se ha elegido el archivo a leer, el tiempo que tarda la aplicación en extraer el mensaje y descifrarlo parece inmediato a los ojos humanos.

### 7.2.1 GENERACIÓN DE UN CÓDIGO QR

Como el código QR sólo se genera con un mensaje no cifrado y, salvo éste, no influye nada más en su generación, es lógico pensar que el tiempo de generación de un código QR es directamente proporcional al número de caracteres del mensaje.

Como esto es así, se han realizado varias pruebas consistentes en la generación de código QR a partir de un mensaje de longitud variable. A continuación, se muestran los resultados obtenidos:

| LONGITUD DEL MENSAJE | TIEMPO |
|---|---|
| **100 caracteres** | 0,825 segundos |
| **500 caracteres** | 0,952 segundos |
| **1000 caracteres** | 1,060 segundos |
| **1500 caracteres** | 1,103 segundos |
| **2000 caracteres (máximo)** | 1,176 segundos |

Tabla 131: Tiempo de generación QR

Como se puede observar, en el peor de los casos, la aplicación tardará poco más de 1 segundo en generar el código QR. Se complementa esta información con una pequeña gráfica para observar que la diferencia temporal cuanto más caracteres componen el mensaje es cada vez menor.

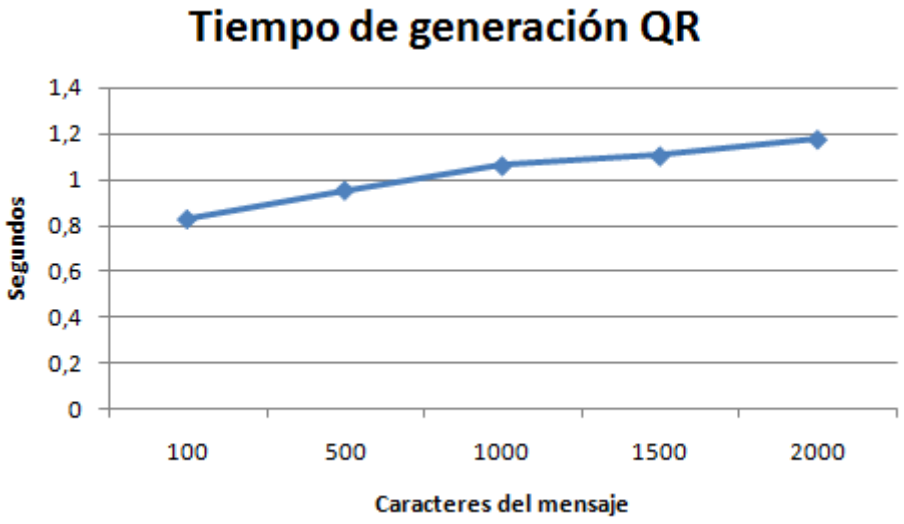



Ilustración 78: Gráfica de tiempo de generación QR

Se considera que el tiempo máximo que tarda la aplicación en generar un código QR es muy bajo (poco más de 1 segundo), lo que quiere decir que se ha cumplido con el requisito de usuario RUR-07 en el que se exigía que la aplicación debía tardar un máximo de 30 segundos en generar un código de esta naturaleza.

### 7.2.2 GENERACIÓN DE UN CÓDIGO QRYPT

Al contrario que el código QR, el mensaje que se quiere incluir en el código QRypt está cifrado; por lo tanto, estas pruebas abarcan el proceso de cifrado y de generación del código. Como en el capítulo anterior se ha analizado lo que tarda la aplicación en generar un código QR a partir de un tamaño variable de mensaje y eso puede, en este apartado se considerará, 2 tamaños de mensaje para que las comparaciones sean más significativas: 500 y 2000 caracteres.

#### *7.2.2.1 MENSAJE DE 500 CARACTERES*

En este apartado se van a calcular diferentes tiempos de cómputo en la generación de un código QRypt a partir de un mensaje de 500 caracteres utilizando diferentes claves de cifrado.

Como dato importante, el número de caracteres que se van a utilizar en los distintos tipos de cifrado está representado en la siguiente tabla:

| TIPO DE CIFRADO | CARACTERES |
|---|---|
| **Contraseña de 10 caracteres** | 10 |
| **Imei** | 15 |
| **Modelo** | 15 (En este caso: SAMSUNGGT-I9100) |
| **Fecha** | 8 (Año: 4 + Mes: 2 + Día: 2) |
| **Patrón de 6 posiciones** | 6 |

Tabla 132: Número de caracteres utilizados en los tipos de cifrado

A continuación, se muestran los distintos tiempos que ha demorado la aplicación en generar un código QRypt cifrado con los diferentes tipos de cifrado:

| TIPO DE CIFRADO | TIEMPO |
|---|---|
| **Contraseña de 10 caracteres** | 1,982 segundos |
| **Imei** | 2,345 segundos |
| **Modelo** | 2,513 segundos |
| **Fecha** | 1,564 segundos |
| **Patrón de 6 posiciones** | 1,231 segundos |

Tabla 133: Tiempo de generación QRypt de 500 caracteres

Aunque tenga el mismo número de caracteres el modelo y el imei del terminal (15 caracteres), se puede observar que cifrar con el modelo tarda más. Esto se debe a que, al obtener el imei sólo se hace una llamada al sistema, mientras que, al obtener la marca y el modelo, se hacen dos.

#### *7.2.2.2 MENSAJE DE 2000 CARACTERES*

Después de estos datos, se muestran los tiempos tomados de las mismas pruebas pero, esta vez, con un mensaje de 2000 caracteres:

| TIPO DE CIFRADO | TIEMPO |
|---|---|
| **Contraseña de 10 caracteres** | 2,311 segundos |
| **Imei** | 2,548 segundos |
| **Modelo** | 2,817 segundos |
| **Fecha** | 2,168 segundos |
| **Patrón de 6 posiciones** | 1,862 segundos |

Tabla 134: Tiempo de generación QRypt de 2000 caracteres

#### 7.2.2.3 COMPARACIÓN

Seguidamente y, por último en esta sección, se presentan los datos en forma de gráfica para poder realizar una comparación más cómoda entre las diferencias de rendimiento que presenta la generación de un código QRypt con uno y otro de los tamaños de mensaje:

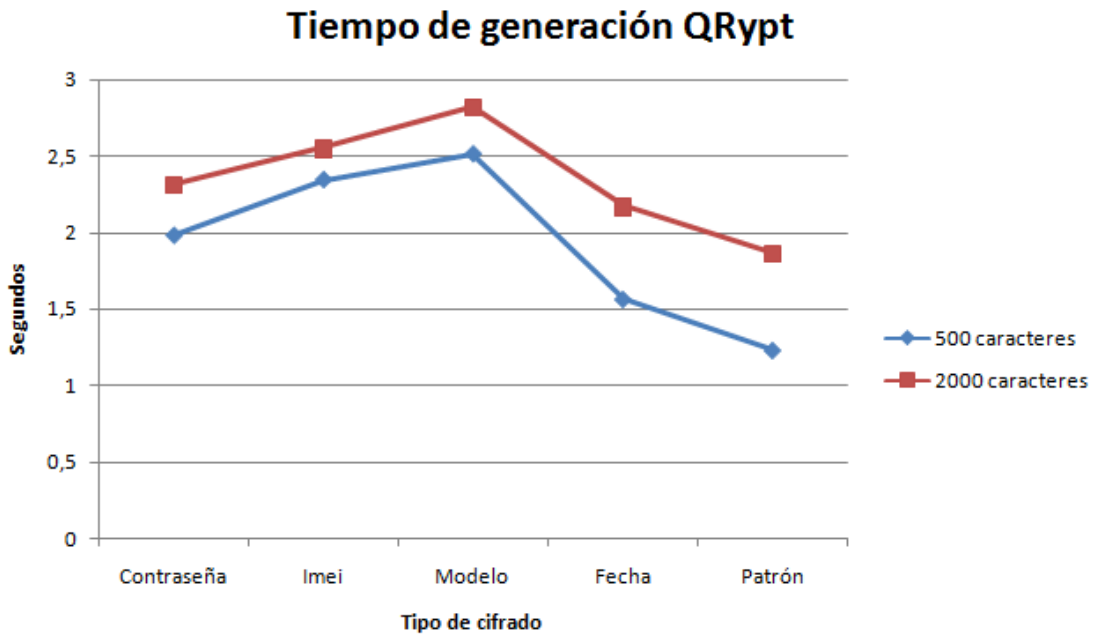



Ilustración 79: Gráfica de tiempo de generación QRypt

Como se puede observar en la ilustración anterior y, como era de esperar, se tarda más tiempo en generar un código QRypt cuanto más grande sea el tamaño de su mensaje y de su clave. Aun así, la diferencia entre cifrar un mensaje de 500 caracteres y uno de 2000 es muy pequeña en razón de tiempo.

Además, cabe resaltar que la aplicación, en el peor de los casos, no llega a tardar ni siquiera 3 segundos en generar un código QRypt; por lo tanto, se determina que cumple el requisito de usuario RUR-07, el cuál exigía que la aplicación debería tardar menos de 1 minuto en generar un código QRypt.

## 7.3 INTEGRIDAD

Hasta ahora, se ha evaluado la funcionalidad de la aplicación y su rendimiento, y el sistema ha logrado sus objetivos. Ahora, llega el turno de evaluar la aplicación llevando a cabo alguna prueba de integridad. En esta sección se buscará poner a prueba al sistema en las situaciones más complicadas y comprobar si las supera correctamente.

### 7.3.1 RECUPERAR EL MENSAJE CON SÓLO UNA PARTE DE CÓDIGO

En teoría, la información introducida dentro de un código QR está clonada de forma que, aunque sólo se muestre una parte de éste, el mensaje puede ser recuperado. A continuación, se realizará una prueba en la que se comprobará si esto se cumple en los códigos generados por la aplicación.

Se ha decidido ocultar una parte del código generado para ver hasta qué punto es capaz la aplicación de recuperar su mensaje. El código elegido para tal fin es uno QRypt ya que, al contener tokens de control, en el caso de que no se pudiese recuperar uno de éstos, sería imposible descifrar el mensaje. Debido a que se considera más complicado recuperar el mensaje de un código QRypt correctamente frente a un código QR, la prueba sólo se realizará con un código de este tipo.

Ilustración 80: Prueba de código tapado 1

En la ilustración anterior se muestra el código QRypt completo sin ninguna alteración. Como era de esperar la aplicación recupera el mensaje "Esto es una prueba 2" sin ningún problema. A continuación, se eliminará una parte de este código para comprobar si aún sigue recuperando el mensaje.

Ilustración 81: Prueba de código tapado 2

Se observa que se ha eliminado una parte de la esquina inferior derecha. Aun así, la aplicación ha sido capaz de recuperar el mensaje completo, aunque tardando 1 segundo más de lo normal. Esto se debe a que tiene que buscar en otros sectores la parte del mensaje que no ha podido recuperar.

Ilustración 82: Prueba de código tapado 3

Se ha decidido ir más allá con la prueba y se ha eliminado un poco más del código. El sector eliminado en este caso es más grande. Aun así, la aplicación ha conseguido recuperar el mensaje correctamente. De nuevo, ha tardado un poco más de tiempo en extraerlo.

Ilustración 83: Prueba de código tapado 4

De nuevo ahondando más en las pruebas, se ha decidido eliminar completamente la fila inferior de datos. A diferencia de en los demás resultados, la aplicación no ha sido capaz de extraer el mensaje contenido en el código. Se ha diagnosticado que, no es que no se haya encontrado un token de control y lo haya extraído mal; simplemente, ni siquiera ha sido capaz de obtener el mensaje.

Como prueba final se partirá del código utilizado en la última prueba en la que sí se pudo extraer el mensaje y se eliminará una parte del código alejada del sector eliminado.

Ilustración 84: Prueba de código tapado 5

Como se puede observar, el sector eliminado está muy alejado del sector que se eliminó en otras pruebas. Aunque esté alejado, la aplicación no ha sido capaz de obtener el mensaje tampoco en este caso.

Como conclusión de la prueba, se sentencia que un código QR es recuperable hasta cierto punto. Está claro que, cuando se elimina o se oculta más de la cuenta, el código queda inutilizable. Además, se intuye con el resultado de la última prueba que la recuperación de información perdida en un código QR solamente es factible cuando la información perdida u ocultada pertenece al mismo sector ya que, si se elimina, además, información de otro sector, el mensaje no se podrá extraer.

Estas pruebas han demostrado que la aplicación no es capaz de extraer mensajes de un código QR muy "mutilado" pero, aun así, si lo extrae, no falla en el descifrado. Esto nos lleva a pensar que no es una carencia de la aplicación ya que, cuando extrae el mensaje, lo descifra sin ningún problema, sino una carencia de los códigos QR; ya que, las mimas pruebas se han realizado con la aplicación "Barcode Scanner" y ha fallado en los mismos casos tardando, incluso más, en los  que conseguía extraer el mensaje.

### 7.3.2 RECUPERAR UN MENSAJE DE 2000 CARACTERES DE UN CÓDIGO

El tamaño máximo de caracteres que permite la aplicación para componer un mensaje es de 2000. Se eligió este valor por representar a una cantidad de caracteres bastante grande y para otorgar al código de cierta flexibilidad cuando el mensaje se cifra ya que agranda y el número máximo de caracteres permitidos es mayor que 4000. Aun así, generar un código aunque sea QR simple con un mensaje de estas dimensiones se hace muy grande y se considera que podría ser difícil de leer.

La siguiente prueba consiste en la generación de un código QR y un código QRypt a partir de un mismo mensaje de 2000 caracteres e intentar extraer su mensaje.

#### *7.3.2.1 CÓDIGO QR*

Para que se entienda cuando se ha especificado anteriormente que un código a partir de un mensaje de 2000 caracteres es de grandes dimensiones, la siguiente imagen muestra el código con el que se realizará esta prueba:

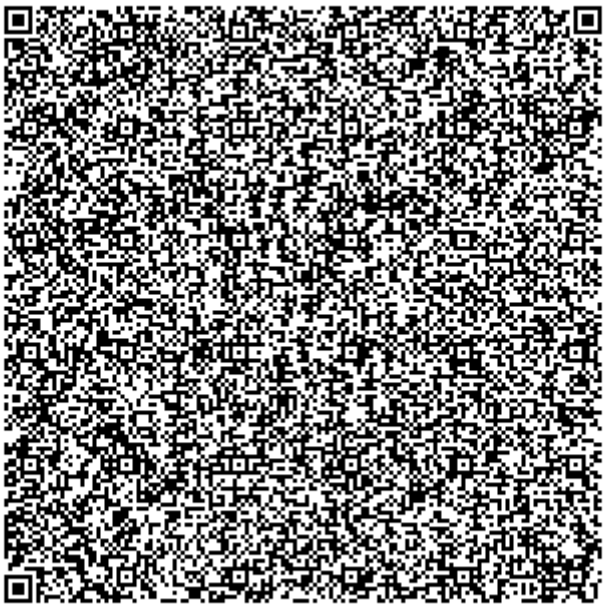


Ilustración 85: Código QR con un mensaje de 2000 caracteres

Como se puede observar, el código está muy comprimido porque contiene muchos datos que, a pesar de que los soporta, quizá sean demasiados.

Cuando se intenta leer este código mediante la cámara del dispositivo, la aplicación es capaz de extraer el mensaje únicamente 4 veces de cada 10; o sea, en un 40% de los casos. Se intuye que esto ocurre debido a que, por razones de luminosidad o pulso del usuario, la cámara nunca es capaz de enfocar completamente el código.

Por otro lado, cuando se intenta leer desde el fichero generado, la aplicación extrae el mensaje correctamente y, además, de forma rápida.

#### 7.3.2.2 CÓDIGO QRYPT

Al igual que se ha hecho en el apartado anterior, se va a mostrar a continuación una imagen con el código QRypt generado a partir de un mensaje de 2000 caracteres y cifrado con el imei des dispositivo:

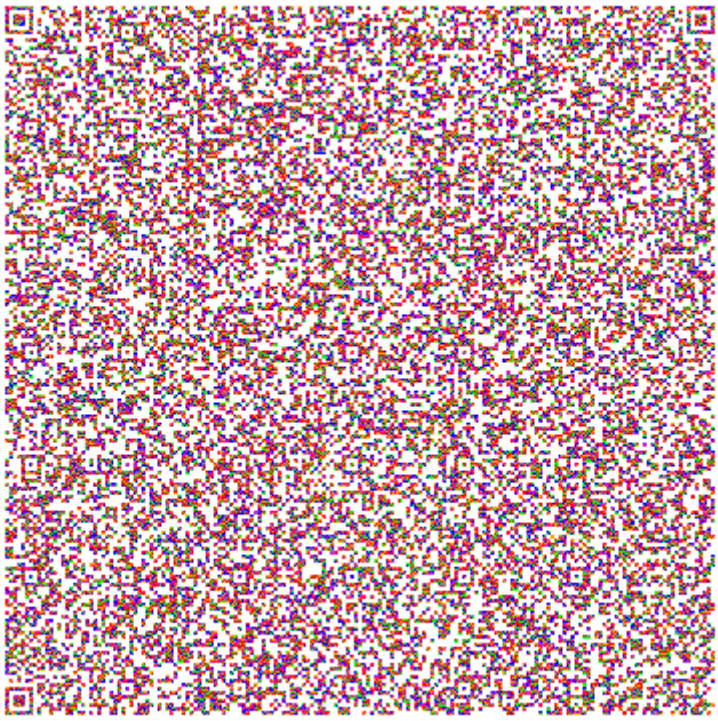


Ilustración 86: Código QRypt con un mensaje de 2000 caracteres

Como se puede observar y al igual que en el anterior caso, el código está muy comprimido porque contiene muchos datos que, a pesar de que los soporta, quizá sean demasiados. Incluso este código contiene más datos que el anterior.

Cuando se intenta leer este código mediante la cámara del dispositivo, la aplicación es capaz de extraer el mensaje únicamente 2 veces de cada 10; o sea, en un 20% de los casos (un 20% menos que en el caso anterior). Se intuye que esto ocurre debido a que, por razones de luminosidad, de colores o pulso del usuario, la cámara nunca es capaz de enfocar completamente el código.

Por otro lado y, como ocurre en el caso del código QR, cuando se intenta leer desde el fichero generado, la aplicación extrae el mensaje correctamente y, además, de forma rápida.

#### *7.3.2.3 CONCLUSIÓN*

Aunque se pueda generar un código QR o QRypt con un mensaje de 2000 caracteres, está claro que quizá sean demasiados. Además, casi nadie escribirá un mensaje tan largo pero, aun así, la aplicación es capaz de leerlo. Lo único que se ha sacado en claro de esta prueba es que la aplicación falla al extraer el mensaje utilizando la cámara del dispositivo pero, aun así, se considera que es un fallo que no puede tener arreglo.

## 7.4 SEGURIDAD

Debido a que el objetivo de este proyecto y la filosofía de la aplicación desarrollada consisten en la seguridad, esta característica de la aplicación no podría quedar sin evaluar.

A continuación, se evaluará la seguridad que ofrece el algoritmo de cifrado que se ha utilizado.

Como ya se ha especificado en otras secciones, el algoritmo de cifrado utilizado para añadir seguridad a la información incluida en el mensaje de un código QRypt es AES con una clave de 256 bits. Como este algoritmo se considera seguro y que no está roto, el único ataque posible que puede realizarse para obtener la información cifrada por él es la fuerza bruta. Este ataque consiste en intentar descifrar el mensaje con todas las combinaciones posibles de la clave hasta que se obtenga la correcta. Como se han utilizado medidas de seguridad adicionales como un *salt* aleatorio que se concatena a la contraseña, un algoritmo de derivación e, incluso, un vector de inicialización, un ataque de estas características se convierte en una medida inviable para un atacante debido a la insuficiencia de recursos computacionales que lo soporten. Como dato adicional, se muestra un tabla que relaciona los distintos tamaños de clave (en bits) con el número de posibles combinaciones que debería realizar un ataque por fuerza bruta: [24]

| TAMAÑO EN BITS DE LA CLAVE | COMBINACIONES POSIBLES |
|---|---|
| **1** | 2 |
| **2** | 4 |
| **4** | 16 |
| **8** | 256 |
| **16** | 65536 |
| **32** | $4.2 \times 10^{9}$ |
| **64** | $1.8 \times 10^{19}$ |
| **128** | $3.4 \times 10^{38}$ |
| **256** | $1.1 \times 10^{77}$ |

Tabla 135: Tabla relacional entre tamaños de clave y posibles combinaciones

Como se puede ver en la tabla anterior, intentar atacar un sistema que haya implementado un algoritmo de cifrado AES con tamaño de clave de 256 bits, significaría tener que generar $1.1 \times 10^{77}$ combinaciones distintas de claves y, además, comprobarlas. Se puede decir que es un ataque completamente inviable ya que, con la potencia computacional de los superordenadores de hoy en día, se tardaría muchos años en generar todas las combinaciones.

Esto demuestra que el algoritmo de cifrado implementado en el sistema es muy seguro pero, aun así, hay un factor que debilita esta seguridad. Este factor es el usuario ya que es el que elige la contraseña y puede que ésta no sea lo suficientemente segura.

Además, cabe resaltar que, por otro lado, la aplicación no almacena aun ningún dato del usuario y que, cuando lo haga (la contraseña por defecto), lo hará en un sector seguro de la aplicación.

# 8 CONCLUSIÓN

Quizá el desarrollo de este Proyecto de Fin de Grado haya resultado en momentos tremendamente exasperante pero, claramente, cuando uno se da cuenta de que ha conseguido lo que se proponía incluyendo funcionalidades adicionales, todos esos momentos de desesperación se quedan en un segundo plano.

Pero, ¿cómo olvidar esos momentos malos cuando han sido responsables de noches sin dormir e, incluso, enfados con amigos y familiares sin motivo alguno? Pues porque gracias a ellos ha sido posible el desarrollo y culminación de este proyecto. Gracias a ellos, no sólo han crecido y crecido las líneas de código de la aplicación. También ha crecido mi experiencia y, lo más importante, he crecido yo como persona.

Después de desarrollar esta aplicación con toda la información previa que tuve que leerme, con todos esos conceptos que tuve que entender y con todas las guías de diseño que tuve que seguir para que la aplicación resultante fuese "un buen trabajo", casi me siento capaz de hacer cualquier cosa.

El estudio de la seguridad y el reconocimiento de patrones siempre han sido una de mis pasiones. Para mí resultó una pena tener que despedirme de las asignaturas relacionadas con la seguridad cuando elegí la especialidad de Computación pero siempre mantuve el interés. Ese interés siguió tan vivo que, gracias a los apuntes tomados por mis amigos que sí cursaron la especialidad de Ingeniería de Computadores y a mis búsquedas compulsivas por Internet, aproveché para basar mi Proyecto de Fin de Grado en eso.

Todo lo que he aprendido durante el desarrollo de este proyecto lo he recibido con ganas y con pasión y aun tengo hambre de más conocimiento. Considero que la especialización en programación móvil está en un momento álgido y que podría incluso dedicarme a ello una vez acabe mis estudios.

Sólo espero que la gente que lea este proyecto, incluso dentro de unos años, disfrute lo mismo que lo que he disfrutado yo realizándolo.

## 8.1 OBJETIVOS ALCANZADOS

Como todo proyecto, éste partió de un objetivo base que se tenía que cumplir. Con el tiempo, este objetivo se ha ido multiplicando, dividiendo, sumando, restando e, incluso, modificando hasta transformarse en la lista completa de objetivos que, a pesar de los problemas, se han alcanzado correctamente.

El objetivo principal consistía en desarrollar una aplicación capaz de generar códigos QR con información que sólo pueda obtener el remitente al que va dirigida. Al final de todo el proceso de desarrollo, los objetivos que se han alcanzado, además del principal, han sido:

- Generación y lectura de códigos QR simples.
- Cifrado de códigos QRypt con 5 tipos de cifrado diferentes.
- Generación de códigos con diferentes colores.
- Redirección cuando el mensaje es una URL.

- Compartición de los códigos generados por medio de muchas aplicaciones de terceros.
- Modificación del idioma de la aplicación.
- Modificación de la ruta de la carpeta de salida.
- Log con las novedades de la versión actual.
- Interfaz simple y accesible para personas con problemas de daltonismo.
- Capas de seguridad adicionales para mejorar el cifrado del mensaje.
- La seguridad de cifrado del mensaje se considera buena.
- El rendimiento de la aplicación en un dispositivo de potencia media ha sido muy bueno.

Después de hacer balance y observar la lista anterior, se puede concluir que se han alcanzado todos los objetivos propuestos durante el desarrollo de la aplicación.

## 8.2 TRABAJOS FUTUROS

Aunque se hayan cumplido todos los objetivos, mientras se desarrollaba la aplicación se han ideado nuevas propuestas que podrían concebirse como trabajos futuros que completen aun más la aplicación.

El primero de estos trabajos es, quizá, el más importante y con el que más apetece ponerse manos a la obra. Se considera que la aplicación es muy completa y su diseño es, además de usable y accesible, bastante original pero de nada sirve realizar una aplicación así si luego no se da a conocer. Por lo tanto, este primer trabajo futuro va a consistir en añadir esta aplicación al repositorio de Android *Google Play* para ponerla en manos de muchos usuarios reales y estudiar su aceptación. Además, de esta forma, los usuarios podrían avisar de problemas que han tenido o proponer nuevas ideas.

Otro trabajo que, debido a su dificultad, no se ha desarrollado en este proyecto es el de generar códigos QR y QRypt con la información distribuida en varios códigos, de forma que, si se leen todos, se obtiene el mensaje completo. Esta funcionalidad permitiría la redacción de mensajes muy extensos e, incluso, funcionaría para actividades o concursos como *gymkanas* en las que habría que leer todos los códigos para obtener el premio.

Quizá otro trabajo interesante debido a su pequeño tamaño, sería poder introducir la información de un archivo Midi en un código de forma que, cuando se lea éste, se reproduzca la canción.

Otro trabajo podría consistir en cifrar un mensaje con el SSID de una red WIFI. De esta forma, el receptor sólo podría leer el contenido del código cuando se encuentre conectado a esa red.

Un trabajo interesante que completaría más la aplicación sería añadir la opción de cifrado "Operador". De esta forma, el mensaje se cifraría con el operador de la tarjeta introducida en el dispositivo generador y sólo podrían acceder a él los usuarios que posean una tarjeta en su terminal del mismo operador. Cabe recalcar que esta novedad sólo estaría presente en los dispositivos móviles que aceptasen tarjeta SIM.

Además, sería muy interesante incluir una funcionalidad consistente en que el usuario pueda cifrar claves WIFI y que, cuando otro usuario lea ese código, se conecte automáticamente su móvil a esa red.

Como se ha podido observar en alguna de las capturas realizadas al dispositivo, hay algunas opciones en el panel de ajustes deslizable que no se han incluido. El acceso a estas funciones se ha diseñado porque desde el principio se ha pensado en ellas. Estas funciones son: guardar datos por defecto del cifrado como la contraseña o el color favoritos (sólo si el usuario quiere), comprobar si existe alguna actualización de la aplicación directamente desde ésta, y visualizar un cuadro de texto con información básica sobre la aplicación (la opción "Acerca de").

Por último, se considera casi indispensable portar la aplicación a iOS. De esta forma el código QRypt podría extenderse abundantemente y, quizá, ser reconocido como estándar. Además, una versión para ordenador de sobremesa de la aplicación resultaría también muy suculenta aunque habría que modificar las librerías para que fuera 100% compatible con Java y modificar la interfaz.

# 9 BIBLIOGRAFÍA

[1] *La Biblia del Programador: Estructura de Android:* http://labibliadelprogramador.blogspot.com.es/2012/09/estructura-de-android.html

[2] *Códigos QR, ¿qué son?, ¿para qué sirven?:* http://suite101.net/article/codigos-qr-que-son-para-que-sirven-a79019

[3] *AndroidCurso.com: Las versiones de Android y niveles de API:* http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/146-las-versiones-de-android-y-niveles-de-api

[4] *GSMArena: Samsung I9100 Galaxy SII:* http://www.gsmarena.com/samsung_i9100_galaxy_s_ii-3621.php

[5] *Diccionario de la lengua española: Intuición:* http://lema.rae.es/drae/?val=intuicion

[6] *Diccionario de la lengua española: Sencillo:* http://lema.rae.es/drae/?val=sencillo

[7] *Themes | Android Developers:* http://developer.android.com/design/style/themes.html

[8] *Feedback | Android Developers:* http://developer.android.com/design/style/touch-feedback.html

[9] *Typography | Android Developers:* http://developer.android.com/design/style/typography.html

[10] *Color | Android Developers:* http://developer.android.com/design/style/color.html

[11] *Iconography | Android Developers:* http://developer.android.com/design/style/iconography.html

[12] *Writing Style | Android Developers:* http://developer.android.com/design/style/writing.html

[13] *Accesibilidad Web: Daltonismo: Problemas de accesibilidad:* http://accesibilidadweb.dlsi.ua.es/?menu=daltonismo-problemas-accesibilidad

[14] *Diccionario de la lengua española: Daltonismo:* http://lema.rae.es/drae/?val=daltonismo

[15] *Web Content Accesibility Guidelines:* http://www.w3.org/TR/WCAG10/

[16] *Web Content Accesibility Guidelines:* http://www.w3.org/TR/WCAG20/

[17] *MINI-GUÍA para evitar problemas a los daltónicos:* http://www.labsk.net/index.php?topic=93403.0

[18] *Ley 32/200., de noviembre, General de Telecomunicaciones:* http://noticias.juridicas.com/base_datos/Admin/l32-2003.html

[19] *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal:* http://noticias.juridicas.com/base_datos/Admin/lo15-1999.html

[20] *zxing - Multi-format 1D/2D barcode image processing library with clients for Android, Java - Google Project Hosting:* https://code.google.com/p/zxing/

[21] *ZBar bar code reader:* http://zbar.sourceforge.net/

[22] *Intent | Android Developers:* http://developer.android.com/reference/android/content/Intent.html

[23] *iGoogle Supported Languages and Countries - iGoogle (Deprecated) - Google Developers:* https://developers.google.com/igoogle/docs/i18n?hl=es

[24] *How secure is AES against brute force attacks?:* http://www.eetimes.com/design/embedded-internet-design/4372428/How-secure-is-AES-against-brute-force-attacks-

[25] *Dudas sobre el código EAN128:* http://www.madisasur.es/EAN128Dudas.aspx

[26] *The Android Source Code | Android Developers:* http://source.android.com/source/index.html

[27] *Diccionario de la lengua española: Seguridad:* http://lema.rae.es/drae/?val=seguridad

[28] *Información sobre la encriptación AES:* http://www.bitzipper.com/es/aes-encryption.html

[29] *AES - Advanced Encryption Standard:* http://computacion.cs.cinvestav.mx/~jjangel/aes/AES_v2005_jjaa.pdf

[30] *RFC 4648 - The Base16, Base32, and Base64 Data Encodings:* https://tools.ietf.org/html/rfc4648#page-5

# 10 ANEXOS

## 10.1 ANEXO A: GLOSARIO DE ACRÓNIMOS Y TÉRMINOS

Esta sección recoge los acrónimos y términos especializados que se han utilizado durante el documento, ordenados alfabéticamente y definidos.

- **ADT (Android Development Tools):** Plugin para el IDE de Eclipse que sirve para poder programar aplicaciones Android mediante varios complementos.
- **API (Application Programming Interface):** Conjunto de funciones y métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- **Bitmap:** Objeto de Android. Es un mapa de bits que representa una imagen.
- **Dropbox:** Servicio que permite el almacenamiento de datos en la nube.
- **Facebook:** Servicio web con naturaleza de red social muy conocido.
- **GHz:** Medida de frecuencia utilizada para determinar la velocidad de cómputo de un procesador.
- **GMail:** Aplicación oficial para dispositivos móviles del servicio de correo homónimo de Google.
- **Google Drive:** Servicio de almacenamiento en la nube semejante a Dropbox desarrollado por Google Inc.
- **Hash:** Cadena que sirve como una representación compacta de una cadena de entrada.
- **IDE (Integrated Development Environment):** Software también llamado "entorno de programación" que está compuesto por herramientas cuyo fin es la realización de software.
- **Imei (International Mobile Equipment Identity):** Código único de los terminales móviles que identifica al dispositivo unívocamente.
- **Integer:** Tipo básico del lenguaje Java que representa a los números enteros en un rango de $2^{32}$ valores.
- **Intent:** Objeto que sirve para invocar componentes o actividades en Android.
- **iOS:** Sistema operativo que emplean los dispositivos móviles de la marca Apple.
- **JPEG (Joint Photographic Experts Group):** Formato de imagen basad en un algoritmo de compresión con pérdidas sobre la imagen original.
- **Kernel:** Núcleo de un sistema operativo. El software responsable de gestionar recursos a través de llamadas al sistema.
- **Line:** Aplicación de mensajería muy famosa para dispositivos móviles y PC.
- **Log:** Equivalente al término "Bitácora". Registro de eventos durante un rango de tiempo.
- **Midi (Musical Instrument Digital Interface):** En el contexto de este proyecto es un tipo de archivo que contiene diversos tipos de datos que representan información musical a nivel de partitura.
- **Píxel:** Menor unidad homogénea de color en una imagen.
- **Plugin:** Equivalente al término "Complemento" en el contexto de los programas informáticos. Expansión que aumenta la funcionalidad de un software.

- **PNG (Portable Network Graphics):** Formato gráfico basado en un algoritmo de compresión para bitmaps.
- **QRypt:** Código QR con información cifrada.
- **RAE (Real Academia Española):** Institución cultural que se dedica a la planificación lingüística mediante la promulgación de normativas.
- **RGB (Red Green Blue):** Modelo de color con el que es posible representar un color mediante la mezcla por adición de los 3 colores primarios.
- **SIM (Subscriber Identity Module):** Tarjeta inteligente desmontable usada en los teléfonos móviles y almacenan de forma segura la clave de servicio del suscriptor usada para identificarse en la red.
- **SSID (Service Set IDentifier):** Nombre de paquete de red inalámbrica que permite identificarla.
- **SQLite:** Sistema de gestión de bases de datos que utiliza el sistema operativo Android.
- **String:** Tipo básico de Java que representa cadenas de caracteres.
- **Superusuario:** Usuario con acceso administrativo.
- **Token:** Cadena de caracteres con un significado determinado.
- **Tweet:** Publicación en forma de mensaje en la página principal del usuario del servicio Twitter.
- **Twitter:** Servicio de blogging que permite enviar mensajes de texto plano de corta longitud que se muestran en la página principal del usuario.
- **URL (Uniform Resource Locator):** Secuencia de caracteres que se usa para localizar e identificar páginas web.
- **Whatsapp:** Aplicación de mensajería muy famosa para dispositivos móviles.
- **Windows:** Sistema operativo creado por Microsoft.

## 10.2 ANEXO B: MANUAL DE USUARIO

### 10.2.1 GENERAR UN CÓDIGO QR

Para generar un código QR con el mensaje deseado, se pulsará en el botón "Generar QR" de la pantalla principal (Ilustración 87).


Ilustración 87: Pantalla principal de la aplicación (Sección Generar)

Una vez se muestre la nueva pantalla (Ilustración 88), se deberá introducir el mensaje que se desea introducir en el código en el campo destinado a ello. Este mensaje debe tener un número de caracteres mayor que 0 pero menor que 2000. Una vez introducido, se pulsará en el botón con forma triangular situado en la esquina superior izquierda.


Ilustración 88: Pantalla de introducción de mensaje

Cuando se abra la nueva pantalla (Ilustración 89), el código se habrá generado y se habrá guardado en la carpeta de salida. Además, se podrá ver el código en la misma pantalla.


Ilustración 89: Pantalla de muestra del código generado 1

### 10.2.2 GENERAR UN CÓDIGO QRYPT

El proceso de generación de un código QRypt es muy parecido al utilizado en la generación QR.

En la pantalla principal (Ilustración 87), se deberá pulsar sobre el botón "Generar QRypt" y, cuando aparezca la nueva pantalla (Ilustración 88), se deberá escribir el mensaje que se quiera introducir en el código.

Después aparecerá una ventana (Ilustración 90) en la que se deberá elegir el/los tipo/s de cifrado que se va/n a utilizar para asegurar el mensaje. Una vez se haya/n seleccionado, se pulsará el botón "Confirmar".


Ilustración 90: Ventana de selección de tipo de cifrado

Cuando aparezca la nueva ventana (Ilustración 91), se deberá escoger el color con el que se quiere tintar el código QRypt que se va a generar. Una vez se ha escogido, se pulsará sobre el botón "Aceptar".

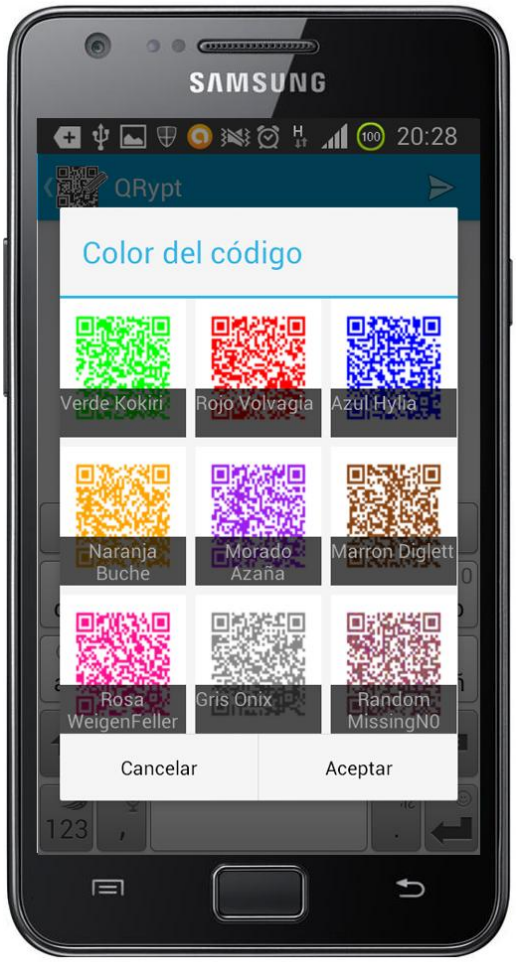



Ilustración 91: Ventana de selección de color

Cuando se abra la nueva pantalla (Ilustración 92), el código se habrá generado y se habrá guardado en la carpeta de salida. Además, se podrá ver el código en la misma pantalla.


Ilustración 92: Pantalla de muestra del código generado 2

### 10.2.3 ENVIAR UN CÓDIGO GENERADO A OTRA APLICACIÓN

Cuando se ha generado un código QR o QRypt, es posible enviar éste a otra aplicación instalada en el dispositivo móvil. Para ello, cuando se abra la pantalla de muestra del código (Ilustración 89 o Ilustración 92), se deberá pulsar sobre el botón "Compartir" de la esquina superior izquierda. Cuando se abra el menú desplegable (Ilustración 93), sólo habrá que elegir la aplicación a la que se quiere enviar.

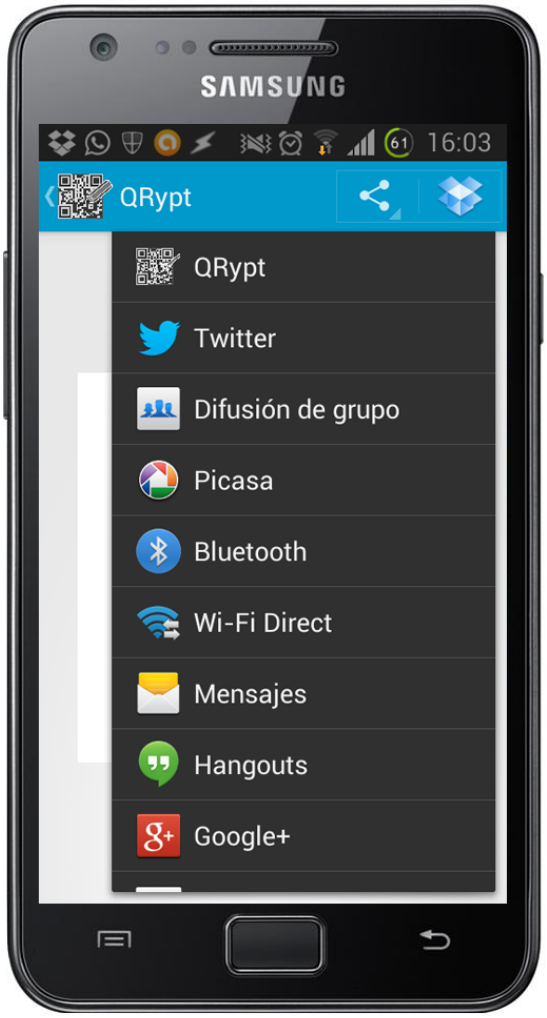



Ilustración 93: Menú desplegable de "Compartir"

### 10.2.4 LEER UN CÓDIGO UTILIZANDO LA CÁMARA DEL DISPOSITIVO

Para leer el mensaje que contiene un código QR o QRypt utilizando la cámara del dispositivo, únicamente se deberá pulsar el botón "Leer de cámara" de la pantalla principal (Ilustración 94).


Ilustración 94: Pantalla principal de la aplicación (Sección Leer)

Una vez se abra la nueva pantalla (Ilustración 95), se deberá enfocar el código que se desea leer hasta que la aplicación lo detecte y abra la siguiente pantalla.

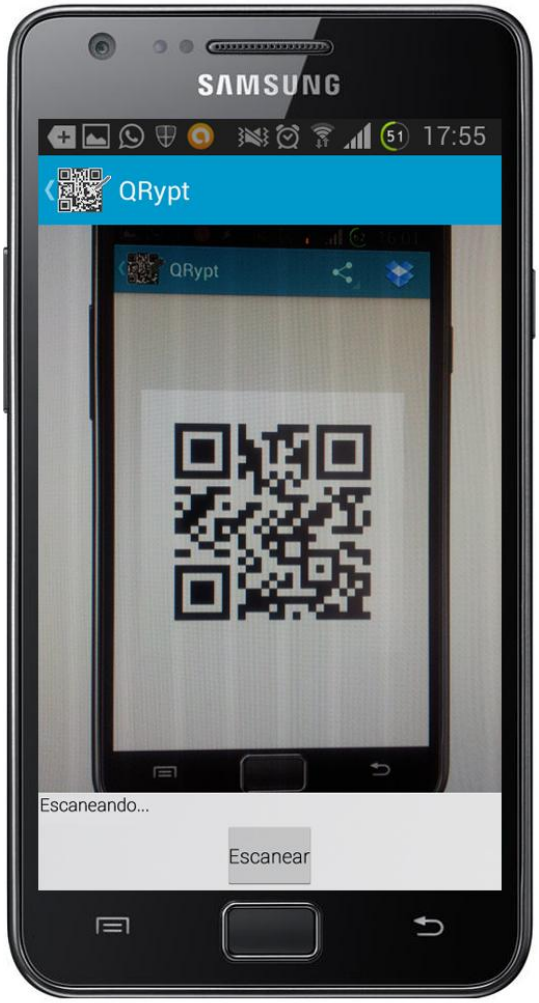



Ilustración 95: Pantalla de lectura por cámara

Una vez la aplicación ha detectado y ha leído el código, se abrirá una nueva pantalla (Ilustración 96) en la que se mostrará el mensaje que éste contenía.


Ilustración 96: Pantalla de muestra del mensaje

### 10.2.5 LEER UN CÓDIGO DESDE UN ARCHIVO

#### *10.2.5.1 OPCIÓN A*

Para extraer el mensaje de un código QR o QRypt que se encuentra en un archivo de imagen, se deberá pulsar sobre el botón "Leer de fichero" en la pantalla principal (Ilustración 94).

Una vez se pulse, se deberá seleccionar el explorador de archivos favorito desde el menú que acabará de aparecer (Ilustración 97).

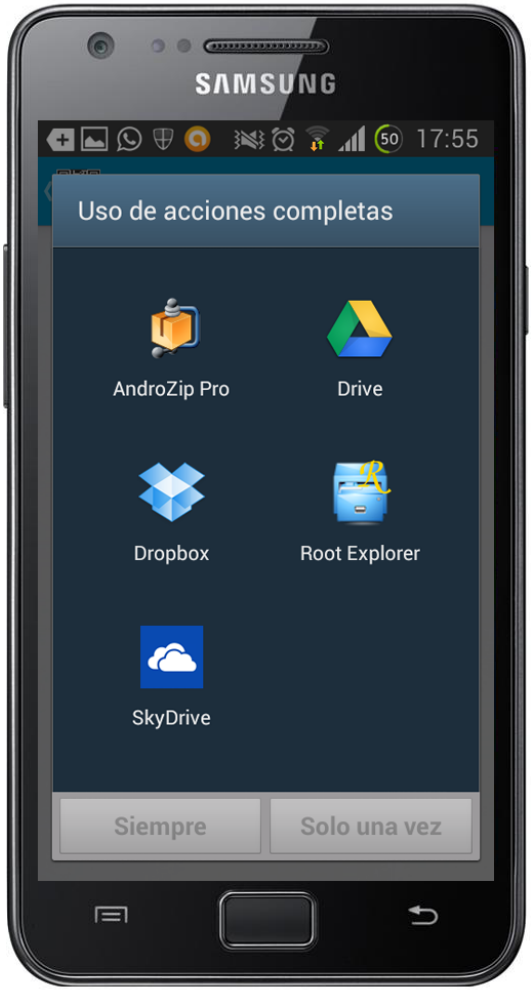



Ilustración 97: Menú de selección de explorador de archivos

Cuando se abra la aplicación seleccionada, se deberá buscar el archivo con el código que se desea leer. Una vez se seleccione, la aplicación abrirá la pantalla de muestra de mensaje (Ilustración 96) en la que se podrá leer la información que contenía el código seleccionado.

NOTA: Si el código estuviese cifrado con una contraseña o con un patrón, se deberá introducir la información pertinente cuando la aplicación lo sugiera.

#### *10.2.5.2 OPCIÓN B*

Otra forma de leer un código contenido en un archivo de imagen es buscarlo directamente en cualquier explorador de archivos o desde la galería y seleccionar la opción de compartirlo con otra aplicación. En el menú que aparezca (Ilustración 98), se deberá elegir la aplicación QRypt para extraer el mensaje del código.

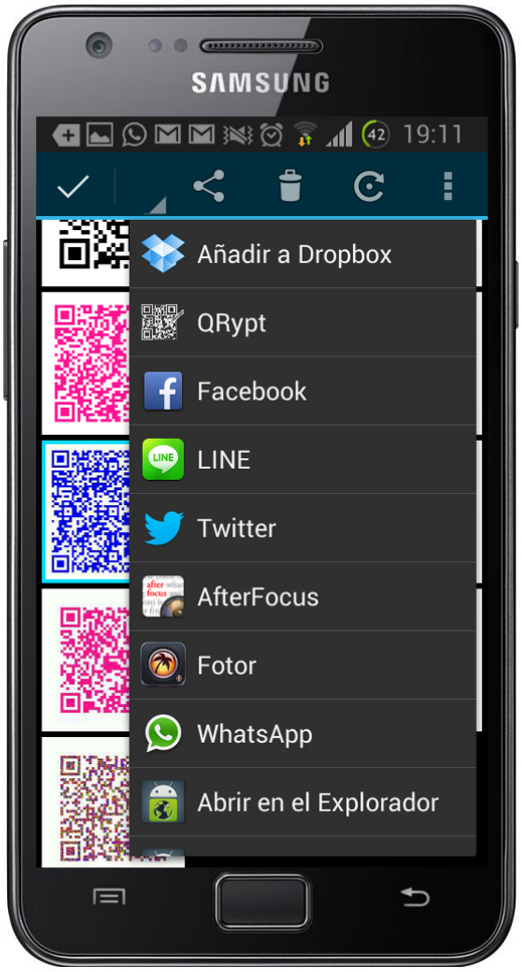



Ilustración 98: Menú de selección de aplicación a la que compartir el archivo

NOTA: Esta funcionalidad no sólo se limita a los exploradores de archivos. Si, por ejemplo, se hubiese obtenido la imagen por una aplicación de mensajería (como Whatsapp o Line), sólo se deberá pulsar en el botón de compartirla y elegir la aplicación QRypt de la lista, tal y como se muestra en la ilustración anterior.

### 10.2.6 CAMBIAR LA RUTA DE LA CARPETA DE SALIDA

Para modificar la ruta de la carpeta de salida donde se guardarán los archivos de imagen con los códigos QR y QRypt generados por la aplicación, sólo se deberá abrir el panel de ajustes (Ilustración 99) deslizando el dedo de izquierda a derecha desde cualquier pantalla o pulsando el botón físico "Menú" del terminal, y seleccionar la opción "Carpeta de salida".


Ilustración 99: Panel de ajustes deslizable

Cuando aparezca la nueva ventana (Ilustración 100), se deberá escribir en el cuadro de texto la ruta que se desee y pulsar el botón "Aceptar".

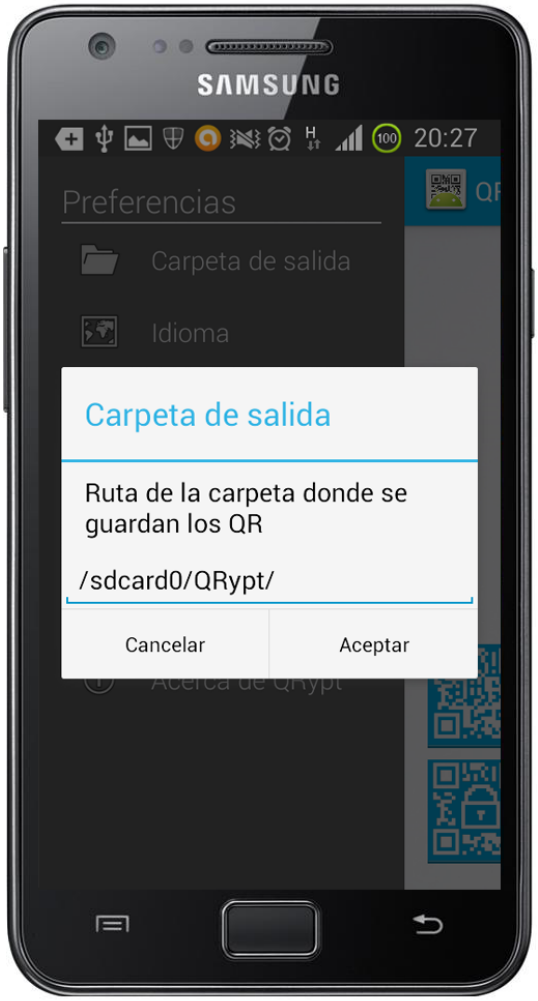



Ilustración 100: Ventana de selección de carpeta de salida

### 10.2.7 CAMBIAR EL IDIOMA DE LA APLICACIÓN

Para modificar el idioma base en el que se muestran los textos de la aplicación, se deberá abrir el panel de justes (Ilustración 99) deslizando el dedo de izquierda a derecha desde cualquier pantalla o pulsando el botón físico "Menú" del terminal, y seleccionar la opción "Idioma".

Cuando aparezca la nueva ventana (Ilustración 101), se deberá elegir el idioma que se quiera y, después, pulsar el botón "Aceptar".

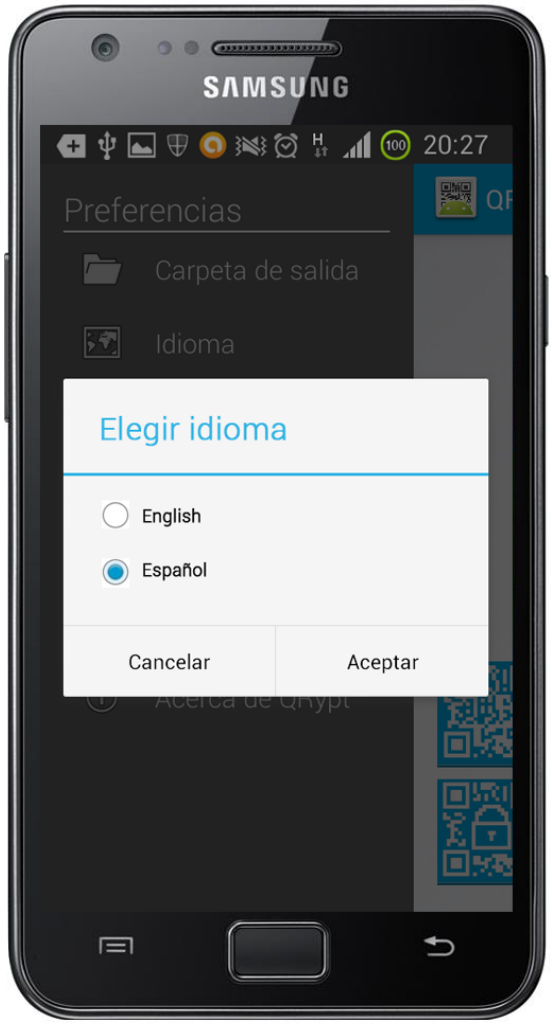



Ilustración 101: Ventana de selección de idioma

### 10.2.8 VISUALIZAR LAS NOVEDADES DE LA VERSIÓN

Para visualizar las novedades que incluye la versión de la aplicación instalada en el dispositivo frente a las demás, se deberá abrir el panel de justes (Ilustración 99) deslizando el dedo de izquierda a derecha desde cualquier pantalla o pulsando el botón físico "Menú" del terminal, y seleccionar la opción "Novedades de la versión".

En la nueva ventana (Ilustración 102) se podrán visualizar dichos datos.


Ilustración 102: Ventana de visualización de las novedades de la versión

## 10.3 ANEXO C: SITUACIONES ÚTILES PARA EL USO DE LA APLICACIÓN

Muchas de las funciones que tiene la aplicación se han implementado pensando en situaciones en las que serían útiles. En este anexo se muestran algunos ejemplos de ellas.

### 10.3.1 HACER PUBLICIDAD

Con la generación de un código QR simple (sin cifrar) se puede introducir información que se quiera que la gente lea. Con la funcionalidad de enviar los códigos generados directamente a aplicaciones, se puede obtener fácilmente el código QR en un equipo con impresora y, tras imprimirlo, colocarlo en alguna zona bien escogida que se sepa que frecuenta mucha gente.

### 10.3.2 COMUNICACIONES SECRETAS

La generación y lectura de códigos QRypt otorga una seguridad mayor que la que muchas aplicaciones de mensajería tienen. Si se quisiese utilizar, por ejemplo, Whatsapp (que no tiene buena fama respecto a su seguridad) para comentar con alguna persona una información muy privada, podría hacerse generando un código QRypt y enviándolo con la opción de compartir a Whatsapp.

### 10.3.3 GUARDAR CONTRASEÑAS

Si el usuario es olvidadizo y nunca recuerda las contraseña (como, por ejemplo, la del correo o la de la cuenta bancaria), puede generar un QRypt con éstas cifradas con el imei de su dispositivo. De esta forma, esas contraseñas sólo se podrán leer si se posee el código y el terminal que lo cifró. Además, siempre se puede utilizar más de un método de cifrado y utilizar, además, una contraseña para este fin.

### 10.3.4 PROMOCIONES

Al contar la aplicación con el tipo de cifrado por fecha, se puede generar códigos QRypt que sólo se puedan leer a partir de una fecha exacta. Se pueden crear muchos códigos que estén cifrados con una fecha exacta en la que, a partir de ella, se deberían poder leer sus datos para alguna promoción. De esta forma, se pueden preparar los códigos mucho antes de la fecha de inicio de la promoción.

### 10.3.5 PROMOCIONES 2

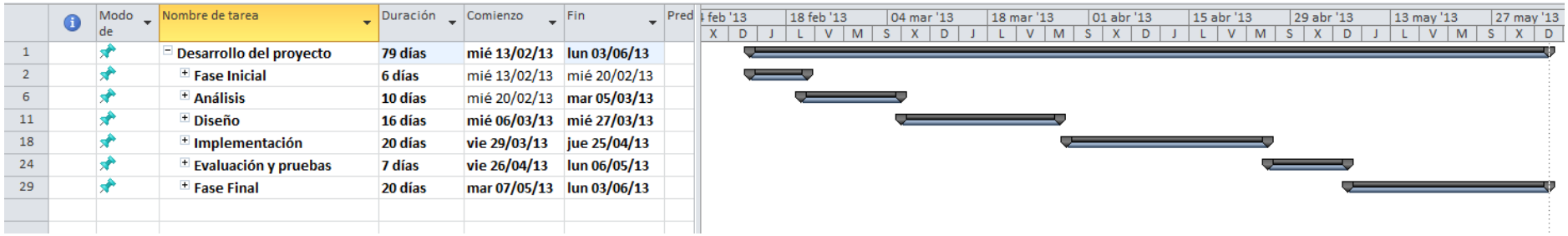
Además, al contar con la opción "Modelo" como tipo de cifrado, las promociones también se pueden dirigir sólo a los usuarios que utilicen un móvil en especial.

# 11 ANEXO D: PLANIFICACIÓN Y PRESUPUESTO

## 11.1 PLANIFICACIÓN

### 11.1.1 PLANIFICACIÓN INICIAL

En esta primera ilustración, se muestra la planificación que inicialmente se estimó. En esta planificación, sólo se incluyen los procesos principales:



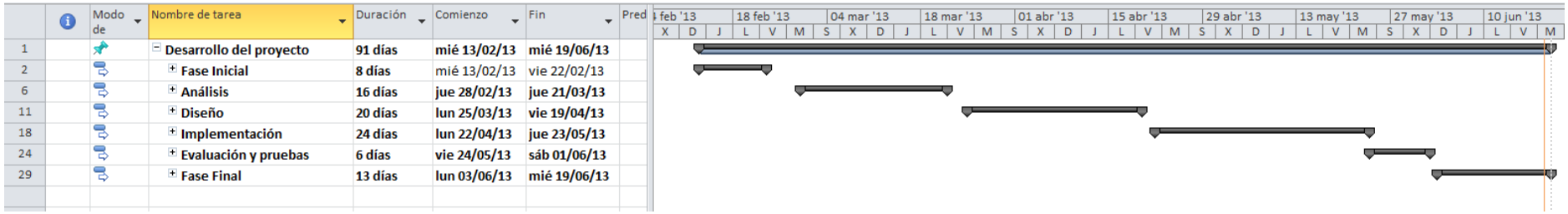
| | Modo de | Nombre de tarea | Duración | Comienzo | Fin | Pred |
|---|---|---|---|---|---|---|
| 1 | | Desarrollo del proyecto | 79 días | mié 13/02/13 | lun 03/06/13 | |
| 2 | | Fase Inicial | 6 días | mié 13/02/13 | mié 20/02/13 | |
| 6 | | Análisis | 10 días | mié 20/02/13 | mar 05/03/13 | |
| 11 | | Diseño | 16 días | mié 06/03/13 | mié 27/03/13 | |
| 18 | | Implementación | 20 días | vie 29/03/13 | jue 25/04/13 | |
| 24 | | Evaluación y pruebas | 7 días | vie 26/04/13 | lun 06/05/13 | |
| 29 | | Fase Final | 20 días | mar 07/05/13 | lun 03/06/13 | |



Ilustración 103: Diagrama de Gantt de la planificación inicial simplificado

### 11.1.2 PLANIFICACIÓN FINAL

Además de esta planificación inicial, se representa a continuación cómo fue finalmente dicha planificación:

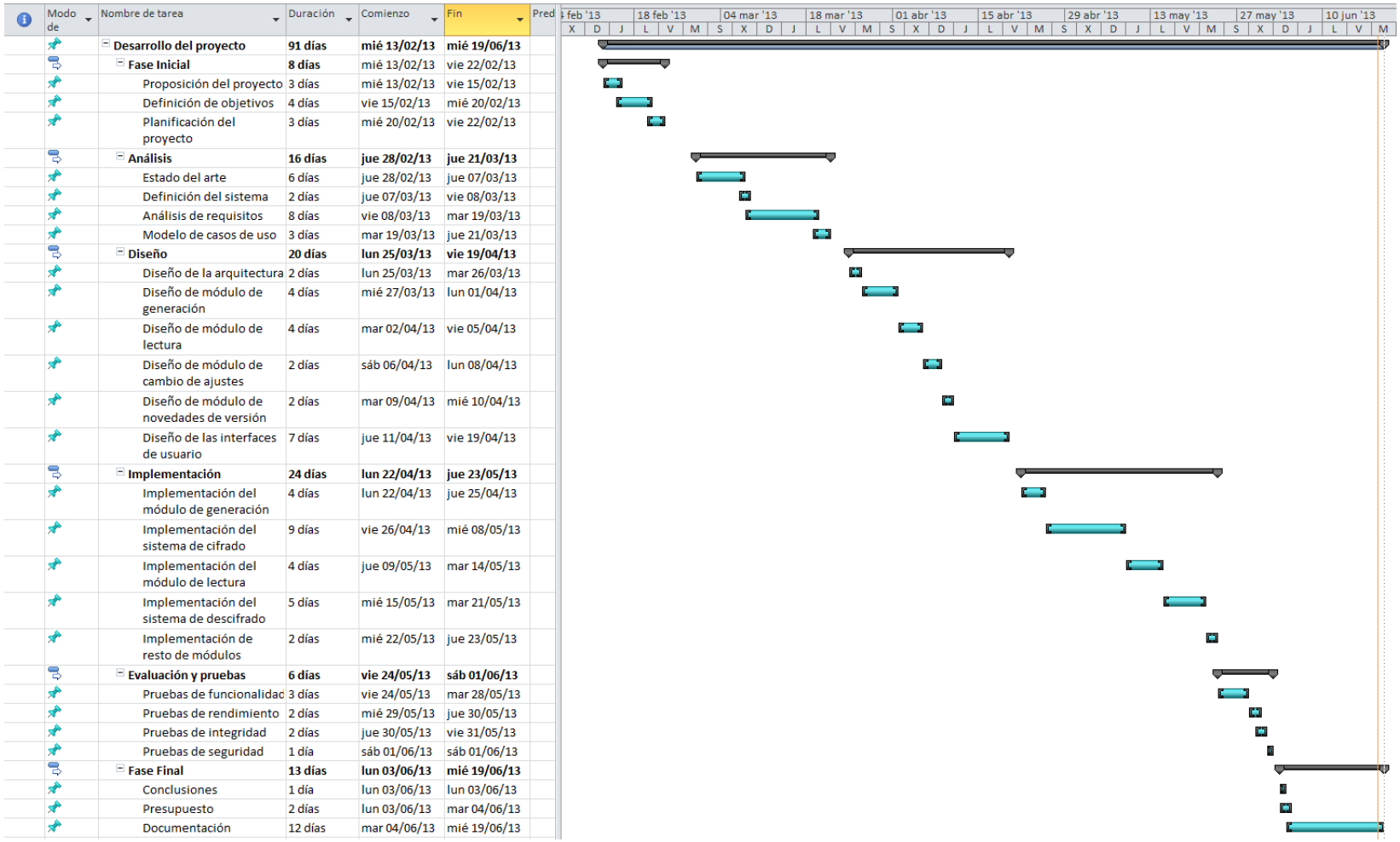


| | Modo de | Nombre de tarea | Duración | Comienzo | Fin | Pred |
|---|---|---|---|---|---|---|
| 1 | | Desarrollo del proyecto | 91 días | mié 13/02/13 | mié 19/06/13 | |
| 2 | | Fase Inicial | 8 días | mié 13/02/13 | vie 22/02/13 | |
| 6 | | Análisis | 16 días | jue 28/02/13 | jue 21/03/13 | |
| 11 | | Diseño | 20 días | lun 25/03/13 | vie 19/04/13 | |
| 18 | | Implementación | 24 días | lun 22/04/13 | jue 23/05/13 | |
| 24 | | Evaluación y pruebas | 6 días | vie 24/05/13 | sáb 01/06/13 | |
| 29 | | Fase Final | 13 días | lun 03/06/13 | mié 19/06/13 | |



Ilustración 104: Diagrama de Gantt de la planificación final simplificado

Por último, se muestra más en detalle esta planificación final:



| Modo de | Nombre de tarea | Duración | Comienzo | Fin | Pred |
|---|---|---|---|---|---|
| | **Desarrollo del proyecto** | **91 días** | **mié 13/02/13** | **mié 19/06/13** | |
| | **Fase Inicial** | **8 días** | mié 13/02/13 | vie 22/02/13 | |
| | Proposición del proyecto | 3 días | mié 13/02/13 | vie 15/02/13 | |
| | Definición de objetivos | 4 días | vie 15/02/13 | mié 20/02/13 | |
| | Planificación del proyecto | 3 días | mié 20/02/13 | vie 22/02/13 | |
| | **Análisis** | **16 días** | **jue 28/02/13** | **jue 21/03/13** | |
| | Estado del arte | 6 días | jue 28/02/13 | jue 07/03/13 | |
| | Definición del sistema | 2 días | jue 07/03/13 | vie 08/03/13 | |
| | Análisis de requisitos | 8 días | vie 08/03/13 | mar 19/03/13 | |
| | Modelo de casos de uso | 3 días | mar 19/03/13 | jue 21/03/13 | |
| | **Diseño** | **20 días** | **lun 25/03/13** | **vie 19/04/13** | |
| | Diseño de la arquitectura | 2 días | lun 25/03/13 | mar 26/03/13 | |
| | Diseño de módulo de generación | 4 días | mié 27/03/13 | lun 01/04/13 | |
| | Diseño de módulo de lectura | 4 días | mar 02/04/13 | vie 05/04/13 | |
| | Diseño de módulo de cambio de ajustes | 2 días | sáb 06/04/13 | lun 08/04/13 | |
| | Diseño de módulo de novedades de versión | 2 días | mar 09/04/13 | mié 10/04/13 | |
| | Diseño de las interfaces de usuario | 7 días | jue 11/04/13 | vie 19/04/13 | |
| | **Implementación** | **24 días** | **lun 22/04/13** | **jue 23/05/13** | |
| | Implementación del módulo de generación | 4 días | lun 22/04/13 | jue 25/04/13 | |
| | Implementación del sistema de cifrado | 9 días | vie 26/04/13 | mié 08/05/13 | |
| | Implementación del módulo de lectura | 4 días | jue 09/05/13 | mar 14/05/13 | |
| | Implementación del sistema de descifrado | 5 días | mié 15/05/13 | mar 21/05/13 | |
| | Implementación de resto de módulos | 2 días | mié 22/05/13 | jue 23/05/13 | |
| | **Evaluación y pruebas** | **6 días** | **vie 24/05/13** | **sáb 01/06/13** | |
| | Pruebas de funcionalidad | 3 días | vie 24/05/13 | mar 28/05/13 | |
| | Pruebas de rendimiento | 2 días | mié 29/05/13 | jue 30/05/13 | |
| | Pruebas de integridad | 2 días | jue 30/05/13 | vie 31/05/13 | |
| | Pruebas de seguridad | 1 día | sáb 01/06/13 | sáb 01/06/13 | |
| | **Fase Final** | **13 días** | **lun 03/06/13** | **mié 19/06/13** | |
| | Conclusiones | 1 día | lun 03/06/13 | lun 03/06/13 | |
| | Presupuesto | 2 días | lun 03/06/13 | mar 04/06/13 | |
| | Documentación | 12 días | mar 04/06/13 | mié 19/06/13 | |



Ilustración 105: Diagrama de Gantt de la planificación final extendido

### 11.1.3 CONCLUSIONES

Si se comparan los diagramas, se llega a la conclusión de que, como era de esperar, la planificación inicial predijo que se tardarían menos días en desarrollar el proyecto. De todas formas, sólo hay una diferencia de 12 días debido a que, a pesar de que se predijo que se tardaría menos en realizar la mayoría de las tareas, se pensó que la documentación llevaría mucho más tiempo (19 días en vez de 12).

La principal desviación se observa que se produce en las tres secciones principales: el análisis del sistema, el diseño, y la implementación. Los motivos por los que ocurrió esto se exponen seguidamente:

- El tiempo que se invirtió en la investigación fue mayor del previsto.
- No se predijo la necesidad de modificar las librerías principales de tratamiento de códigos QR para que funcionase en Android.
- La experiencia en el desarrollo de aplicaciones móviles para el sistema operativo Android era casi nula.

A pesar de la diferencia de días, se ha logrado desarrollar una aplicación con más funcionalidades que las pensadas en un principio.

## 11.2 PRESUPUESTO

1. **Autor:** Carlos Pliego García
2. **Departamento:** Departamento de Informática
3. **Descripción del proyecto:**
    a. **Título:** Desarrollo de una aplicación generadora y lectora de códigos QR seguros en Android
    b. **Duración (meses):** 4
    c. **Tasa de costes indirectos:** 18%
4. **Presupuesto total del proyecto (valores en Euros):** 17575,23.
5. **Desglose presupuestario:**

| PERSONAL | | | | |
|---|---|---|---|---|
| **Apellidos, Nombre** | **Categoría** | **Horas de trabajo** | **Coste por hora (€)** | **Coste Total (€)** |
| **Pliego García, Carlos** | Jefe de proyecto | 88 | 30,00 | 2640 |
| **Pliego García, Carlos** | Diseñador | 160 | 18,00 | 2880 |
| **Pliego García, Carlos** | Analista | 128 | 20,00 | 2560 |
| **Pliego García, Carlos** | Programador | 192 | 14,00 | 2688 |
| **Pliego García, Carlos** | Encargado de pruebas | 48 | 10,00 | 480 |
| **Pliego García, Carlos** | Documentalista | 96 | 18,00 | 1728 |
| | | | **Total (€)** | **12976** |

Tabla 136: Coste del personal

| EQUIPOS | | | | | |
|---|---|---|---|---|---|
| **Descripción** | **Coste (€)** | **Uso dedicado al proyecto (%)** | **Dedicación (meses)** | **Período de depreciación** | **Coste imputable (€)** |
| **Samsung Galaxy SII (GT-I9100)** | 400 | 100 | 4 | 60 | 23 |
| **PC de sobremesa (a piezas)** | 850 | 100 | 4 | 60 | 47 |
| **Asus A55A-SX465H** | 600 | 100 | 4 | 60 | 33 |
| | | | | **Total (€)** | **103** |

Tabla 137: Coste de los equipos

| SOFTWARE | | | | | |
|---|---|---|---|---|---|
| **Descripción** | **Coste (€)** | **Uso dedicado al proyecto (%)** | **Dedicación (meses)** | **Período de depreciación** | **Coste imputable (€)** |
| **Licencia de Windows 8 Pro** | 279,99 | 100 | 4 | 60 | 16 |
| **Microsoft Office 2007 Professional** | 709 | 100 | 4 | 50 | 47 |
| **Microsoft Project 2010 Professional** | 729 | 100 | 4 | 50 | 49 |
| **Adobe Photoshop CS4** | 300 | 100 | 4 | 50 | 20 |
| | | | | **Total (€)** | **132** |

Tabla 138: Coste del software

| OTROS COSTES DEL PROYECTO | | |
|---|---|---|
| **Descripción** | **Empresa** | **Costes imputable (€)** |
| **Internet** | Telefónica | 48 |
| **Luz** | Iberdrola | 40 |
| | **Total (€)** | **88** |

Tabla 139: Otros costes

## 6. Resumen de costes:

| Concepto | Coste (€) |
|---|---|
| **Personal** | 12976 |
| **Equipos** | 103 |
| **Software** | 132 |
| **Otros** | 88 |
| **Costes indirectos** | 1225,98 |
| **TOTAL SIN IVA** | **14524,98** |
| **IVA (21%)** | 3050,25 |
| **TOTAL CON IVA** | **17575,23** |

Tabla 140: Resumen de costes