Prototipo para recomendaciones de seguridad para implementar aplicaciones en dispositivos móviles Android

Miguel Andrés Peñata Rangel

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES
FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
2019

PROTOTIPO PARA RECOMENDACIONES DE SEGURIDAD PARA IMPLEMENTAR APLICACIONES EN DISPOSITIVOS MÓVILES ANDROID

MIGUEL ANDRÉS PEÑATA RANGEL

TRABAJO DE GRADO
PARA OBTENER EL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR
Ing. Celio Gil Aros

CODIRECTOR
Ing. LUIS EDUARDO BAQUERO

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES
FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
2019

NOTA DE ACEPTACIÓN

_____

_____

_____

PRESIDENTE DEL JURADO

_____

_____

JURADO

_____

_____

JURADO

_____

Bogotá, D.C., marzo de 2019

DEDICATORIA

A mi familia, quienes han sido mi mundo. Y a una estrella.

A quienes me han apoyado desde el minuto 1 hasta el 90 y en tiempo extra, quienes me han tendido el cobijo necesario y me han dicho sin cesar que todo en esta vida lo puedes obtener si trabajas duro y crees en ti.

Este gran esfuerzo, casi a contra reloj, que se ha convertido en una de las experiencias más desafiantes y hermosas de mi carrera profesional, lo dedico a las personas que me han tendido una mano, abriéndome una pequeña ventana de oportunidades en forma de conocimiento.

AGRADECIMIENTOS

Por su sabiduría y entendimiento, por su guía y su apoyo emocional y académico, a los docentes Ingeniero Celio Gil Aros, Ingeniero Luis Eduardo Baquero, y demás docentes que hicieron parte de este largo y satisfactorio proceso de formación profesional y personal.

CONTENIDO

Pág.

TABLAS

Pág.

ILUSTRACIONES

Pág.

RESUMEN

Los desarrollos actuales dejan de lado la parte de seguridad, tomándola más como un gasto que como una inversión, comprometiendo la integridad de los principales implicados en el funcionamiento de la aplicación como plataforma de almacenamiento de la información del creador, el administrador y los usuarios.

Este proyecto está diseñado para aplicaciones desarrolladas para plataformas móviles Android. Se llevó a cabo mediante la metodología ágil SCRUM, implementando una base de datos en Firebase que almacena las recomendaciones de seguridad, las cuales serán presentadas al usuario una vez se responde un cuestionario para determinar las características de la aplicación de la cual se desean obtener las recomendaciones de seguridad. La aplicación está codificada en lenguaje JAVA para la API de Android.

Palabras clave: Dispositivos móvil, Android, firebase, SRUM, seguridad informática, sistemas.

INTRODUCCIÓN

Según el estudio de Google “Seguridad del ecosistema Android 2018”, el cual comenzó a ser publicado mensualmente a partir de septiembre de 2018; y que previamente era publicado anualmente como “Informe anual de seguridad”, tenemos información más frecuente sobre los porcentajes de aplicaciones potencialmente dañinas instaladas, en qué versiones de Android y los países donde más la sufren. Según el informe comprendido entre enero de 2017 y septiembre de 2018, la versión *Marshmallow* ha sido la más vulnerable, contando con un 0,665% de dispositivos con aplicaciones potencialmente dañinas. De las cuales, las más comunes son: Troyano (62,9%), fraude por clic (21,2), fraude por sms (4,6%), puerta trasera (4,5%), entre otros.

Adicionalmente, según OWASP; el promedio de aplicaciones consiste de 106 *open source* (código abierto), y típicamente contiene 23 vulnerabilidades conocidas, sumados a la implementación de componentes antiguos, sin soporte e impopulares.

La mayoría de aplicaciones potencialmente dañinas no son desarrolladas de forma directa con ese propósito, sino que contienen fallas en su seguridad que han sido aprovechadas para ser vulneradas para maquillar un propósito ajeno al deseado. Con base a este problema, se plantea la implementación de una aplicación que recoja las características de una aplicación móvil que se desea desarrollar y arroje las recomendaciones de seguridad que ayuden a evitar que el propósito de la misma sea desviado aprovechando una brecha en la seguridad, comenzando desde la política hasta la codificación, convirtiéndola en una aplicación potencialmente dañina con todos los riesgos a la seguridad y seguridad de la información que esto implica.

1. ASPECTOS DE LA INVESTIGACION

1.1 DESCRIPCIÓN DEL PROBLEMA

Al momento de que una persona, microempresa o empresa de gran tamaño, desea llevar su producto al mercado móvil, uno de los apartados a los cuales se le descalifica con mayor facilidad es la seguridad, es vista más como un gasto que como una inversión. Se desea que el manejo de la aplicación sea seguro tanto para el usuario que desea hacer uso del servicio prestado como para quien recibe los beneficios de la aplicación; pero, ¿hasta qué punto un sistema de seguridad es demasiado, o muy poco? Al momento de comenzar un emprendimiento, en este caso el desarrollo de una plataforma móvil, no hay mucho dinero del cual tomar ni se puede arriesgar a perder cualquier beneficio que se pudiese haber obtenido al implementar un software inseguro. Se plantea un problema donde un usuario quiere elegir un sistema de seguridad para su aplicativo, pero no sabe qué tanta o qué tan poca seguridad necesita con base a las características de la aplicación a desarrollar. (OWASP, 2017)

1.2 FORMULACION DE LA PREGUNTA DE INVESTIGACION

¿Cómo a través de una aplicación móvil se puede analizar las características de una aplicación de Android permitiendo notificar al usuario de las vulnerabilidades potenciales, sugerir unas políticas de seguridad y unos métodos para lidiar con las vulnerabilidades?

1.3 JUSTIFICACION DEL PROYECTO DE INVESTIGACION

El desarrollo del presente proyecto permite al estudiante aplicar los conocimientos teóricos aprendidos en el transcurso de su carrera, dentro del marco comprendido por el desarrollo para plataformas móviles de un proyecto con metodología ágil SCRUM.

1.3.1 Razones Sociales

- Facilitar a los nuevos empresarios e impulsadores la accesibilidad a sistemas de seguridad acordes a sus necesidades, y, por ende, fomentar la implementación de sistemas más seguros.

- Presentar al público el conocimiento básico sobre sistemas de seguridad, sus usos más comunes y las vulnerabilidades que afrontan.

- Fomentar la buena praxis para con el desarrollo preliminar de plataformas móviles en cuanto a su seguridad, reduciendo el número de estafas y brindando un poco de luz sobre un tema al cual se le presta poca atención y sobre el cual se debería invertir más; pero, también invertir bien.

1.3.2 Razones Económicas

Que una pequeña o mediana empresa, o persona natural que presente una idea, sepa exactamente qué sistema de seguridad implementar a su plataforma móvil, con ello ahorrando tiempo y dinero adquiriendo el que mejor se ajuste a sus necesidades.

1.3.3 Razones Técnicas

El desarrollo, implementación y maquetación de un sistema de seguridad es un trabajo complejo, más aún si se desea que dicho sistema cumpla ciertos requisitos específicos para aquello a lo cual brindará seguridad. Y dichos recursos de tiempo y dinero no deberían ser invertidos cuando un sistema de seguridad con menos requisitos técnicos sería más apropiado para lidiar con las vulnerabilidades que la plataforma afrontará.

1.4 IMPACTO

Como un aplicativo-plataforma de código libre en ANDROID, su funcionalidad principal sería la aplicación para escenarios reales. Las personas, o pequeñas-medianas empresas, que quieran hacer uso libre de dicha plataforma-guía podrán hacerlo de manera rápida debido a su acceso móvil. De esta forma el conocimiento esparcido por medio de la aplicación podrá llegar a cualquier persona que, aunque no fuese lo que buscase en un primer momento, encontrará en él el valor agregado al conocer cómo y porqué elegir un sistema de seguridad según sus necesidades.

Adicionalmente y debido a su distribución libre, cualquier persona que quiera hacer uso del mismo en un entorno simulado tendrá acceso completo a la base de conocimiento, y recibirá el mismo tipo de recomendaciones por parte de la aplicación al momento de realizar su prueba controlada como si fuese un entorno real.

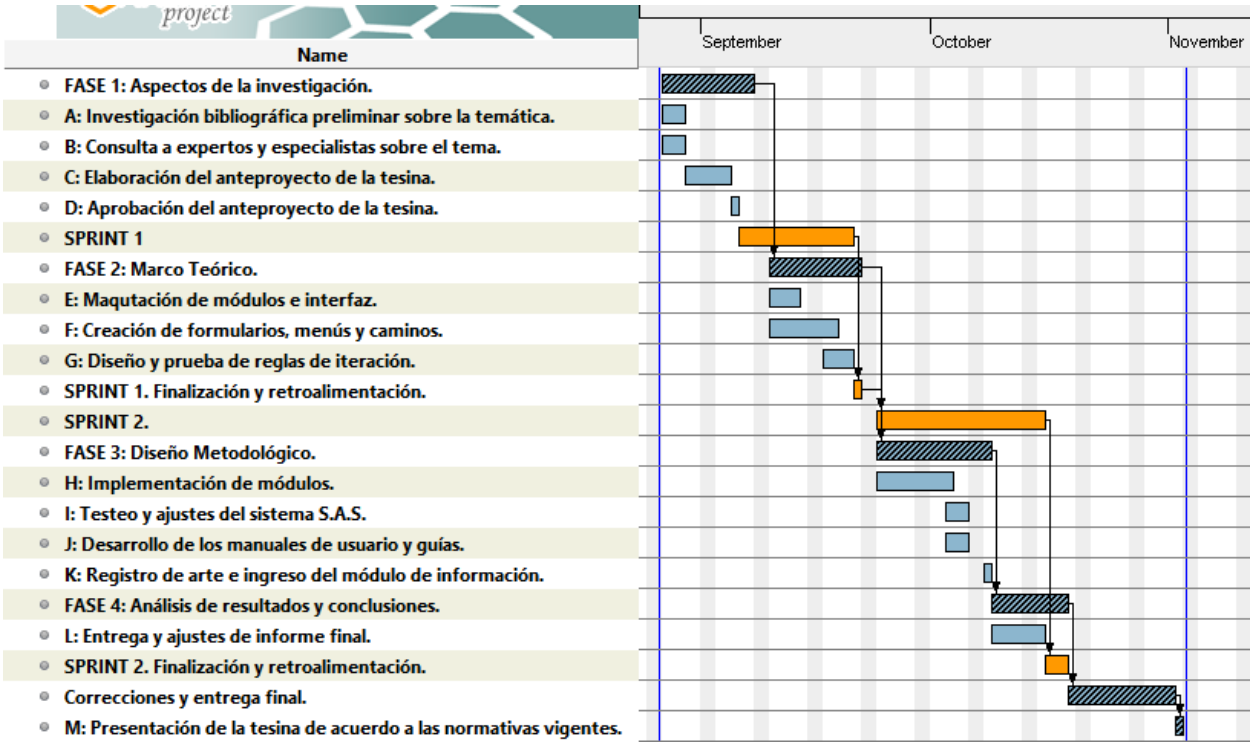
1.5 DELIMITACION

1.6.1 Espacial

- La Fundación Universitaria Los Libertadores, como gestora y facilitadora del proyecto.
- El sistema operativo para entornos móviles ANDROID.

1.6.2 Cronológica

Tabla 1. Duración cronológica del proyecto.



1.7 RECURSOS

1.7.1 Recursos Humanos. Autor del proyecto realizando actividades de análisis, redacción de documentación y anexos, programación en ANDROID, diseño gráfico e implementación y testing de cajas negra y blanca.

Tabla 2. Recurso humano y actividades asignadas.

Recurso humano	Cantidad horas	Costo unitario	Costo total
Análisis de antecedentes (Autor)	35	$15.000	$525.000
Programación Android (Autor)	80	$12.000	$960.000
Diseño e implementación gráfica (Autor)	15	$10.000	$150.000
Pruebas (Autor)	10	$8.000	$80.000
Documentación (Autor)	15	$8.000	$120.000
TOTAL	155	-	$1’835.000

1.7.2 Recursos de hardware. Computadora de alta gama con acceso a programas de desarrollo de software ANDROID, gestores de base de datos y diseño gráfico básico. Lugar de trabajo principal y único, donde se llevarán a cabo todas las tareas de programación, diseño y *testing.*

Tabla 3. Recurso de Hardware y su utilización.

Recurso de hardware	Cantidad	Costo
PC, centro de cómputo usado para la realización de todas	1	$4’000.000

las actividades del proyecto.		
TOTAL	1	$4’000.000

1.7.3 Recursos de software. *Safari Learning Platform.* Recurso bajo subscripción de la organización ACM que se usará para recopilar la información sobre las tecnologías a implementar y la bibliografía sobre la investigación del estado actual del arte. Así como la creación de la base de datos y la base de conocimientos. Subscripción proporcionada por el experto a modo de patrocinio.

Tabla 4. Recursos de software.

Recurso de software	Cantidad	Costo
Android Studio	1	Licencia libre
ArgoUML	1	Licencia libre
Grantt Project	1	Licencia libre
Microsoft Office 365	1	$229.999
Safari Learning Platform Students	1	$179.999
Adobe Photoshop CC	1	$126.560
TOTAL	-	$604.560

1.8 METODOLOGÍA

Debido a las limitaciones de tiempo se ha optado por la metodología de desarrollo ágil SCRUM, metodología donde la prioridad recae sobre la constante retroalimentación de parte del receptor del proyecto, y la necesidad de obtener resultados pronto. Adicionando el hecho de que la innovación, la flexibilidad y la productividad son fundamentales para cumplir con los plazos.

1.9 OBJETIVOS

1.9.1 General

Desarrollar e implementar un prototipo para recomendaciones de seguridad para implementar aplicaciones en dispositivos móviles Android.

1.9.2 Específicos

- Iterar las vulnerabilidades con los sistemas de seguridad que se enfoquen a dichas falencias.
- Implementar un formulario secuencial que tome las distintas posibilidades ingresadas por el usuario en forma de características, arrojando los resultados más adecuados para dichos escenarios.
- Listar una serie de valores preestablecidos que funcionarán como “Mejores elecciones/Elecciones populares” con base en los resultados de las pruebas con las vulnerabilidades y necesidades más comunes.
- Llevar a cabo las distintas etapas de *testing* y depuración de errores para entregar el producto en su etapa 1.0.

2. MARCO TEÓRICO

2.1 ESTADO DEL ARTE

A la fecha de la realización de la presente investigación la plataforma *Play Store* (anteriormente *Android Market*) en su versión estable 0.2.06-all [0] [PR] 197926756 del 25 de mayo de 2018 para dispositivos móviles ANDROID no cuenta con una herramienta, gratuita o de pago, que analice información suministrada por el usuario y suministre un método de aseguramiento personalizado para la plataforma[1].

De acuerdo con dicha evidencia, se estructura el estado del arte con base a dos puntos de enfoque planteados durante la pregunta investigativa, uno enfocado a la seguridad del dispositivo móvil ANDROID, y el otro referido a la seguridad de la transacción financiera llevada a cabo por medio de la plataforma personalizada o por un tercero según sea conveniente. En adición a dos abstractos sobre los antecedentes en la seguridad de control de cuentas de usuario, y en seguridad de la información. Sin tomar en cuenta ni el modelo ni el fabricante del dispositivo.

2.1.1 Antecedentes en seguridad del S.O. ANDROID

Android es un sistema operativo para aplicaciones móviles creado por Google que ha tenido un gran crecimiento desde su lanzamiento en 2008. Construido sobre el kernel de Linux del cual hereda ciertas capacidades de seguridad, además de implementar mecanismos propios que facilitan el desarrollo de aplicaciones. Sin embargo, el hecho de que Android sea una plataforma de código libre y que el desarrollo de aplicaciones para dicho sistema operativo sea tan sencillo, hace que

[1] ©2018 Google, español. Resultados de búsqueda en aplicaciones en todos los idiomas, precios y valoraciones
(https://play.google.com/store/search?q=security&c=apps&hl=es)

construir aplicaciones con propósito malicioso sea fácil. Por ende, los mecanismos actuales de seguridad no son suficientes y la información de los usuarios es vulnerable. (Méndez-Acuña and Sánchez, 2016, p1).

Según ComScore en su estudio anual "2013 Mobile Future in Focus"[2], ANDROID ocupa el primer lugar en ventas con un dominio en el 53 porciento del mercado, seguido por un 36 porciento de dispositivos con Apple iOS.

2.1.1.1 Arquitectura del S.O. ANDROID[3]

El sistema operativo para aplicaciones móviles ANDROID fue desarrollado sobre una modificación del kernel de Linux[4], y cuyos componentes fueron implementados en Java, C, C++ y XML:

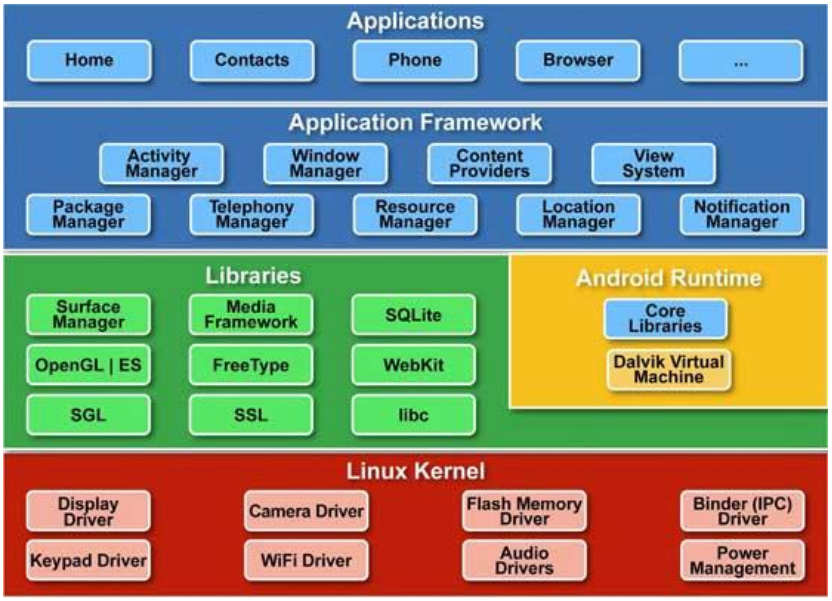
- **Capa de aplicaciones:** Capa superior de la arquitectura del sistema operativo, escrita en lenguaje de programación JAVA. Incluyendo entre sus aplicaciones nativas un programa para SMS, calendario, contactos, navegador (web), entre otras. Las aplicaciones de ANDROID están estrictamente relacionadas, una tarea puede ser vista como una aplicación del usuario. El concepto de tarea es útil en las aplicaciones del SO porque permiten al usuario deshacer o devolver paso a paso operaciones emergentes a modo de pila. Las tareas pueden ser vistas como una serie de actividades posiblemente de múltiples aplicaciones.
- **Framework de aplicaciones:** Este framework proporciona a los desarrolladores de API de las áreas como la interfaz gráfica de usuario, administración de energía, acceso a la red, dispositivos multimedia, búsquedas y acceso a datos. Desde esta misma capa se provee mecanismos para que los componentes del sistema puedan ser reemplazados por el

[2] L. Andrew and A. Carmela. 2013 mobile future in focus, January 2013.
[3] Según J. Park, B. Kim, S.-R. Kim, J. H. Yoo., y Y. Cho. En su Performance analysis of security enforcement on Android operating system. En Proceedings of the 2011 ACM Symposium on Research in Applied Computation, RACS '11, páginas 282-286, New York, NY, USA, 2011. ACM.
[4] S. Brahler. *Analysis of the Android Architecture.* PhD thesis, Karlsruher Institut fur Technologie, 2010.

usuario y la reutilización de los mismos sea simple, las aplicaciones pueden publicar sus funcionalidades y otras pueden utilizarlas de acuerdo a las reglas de seguridad.

- **Librerías:** Las librerías más comunes son las del sistema, la de gráficos y la de medios. Éstas se presentan a los desarrolladores mediante el framework de aplicaciones y están escrita, por lo general, en C y C++. Varía según las distintas versiones de Android.
- **Tiempo de ejecución Android:** Capa intermedia dentro de las librerías que está compuesto en tiempo real por Dalvik, la máquina virtual de Android, y las librerías núcleo. En la máquina virtual es donde se ejecutan las aplicaciones y las librerías núcleo proporcionan las funciones disponibles en las librerías JAVA, debido a que las aplicaciones son escritas en este lenguaje.
- **Kernel de Linux:** Para las funciones básicas del sistema como seguridad, gestión de procesos, memoria y controladores, ANDROID utiliza una modificación de la serie 2.6 del kernel Linux. Uno de los cambios más importantes realizados está en la mejora de la administración de energía, debido a la importancia crítica de este aspecto para el entorno móvil del sistema operativo.

Figura 1. Arquitectura del SO Android.



Fuente: (https://letsknowaboutandroid.wordpress.com/about/, s.f.)

2.1.1.2 Modelo de seguridad de Android

Durante su ciclo más temprano de desarrollo, en sus inicios, el equipo de desarrollo creó un programa enfocado en la seguridad para abordar los puntos débiles de los sistemas operativos móviles. Las principales actividades de seguridad llevadas a cabo por el programa son:

- **Revisión de diseño:** Cada una de las características principales asociadas a la plataforma fue revisada para integrar controles apropiados a la arquitectura del sistema.
- **Pruebas de penetración y revisión del código:** Los componentes del sistema Android de los desarrolladores y aquellos de licencia libre que se utilizaron, pasaron por revisiones detalladas de seguridad llevadas a cabo por el equipo de seguridad de Android, el equipo de seguridad de la información de Google y consultores de seguridad independientes. El objetivo fue, por medio de la simulación de un entorno post-lanzamiento, identificar las debilidades y vulnerabilidades mucho antes de que la plataforma fuera utilizada.

- **Revisión de la comunidad:** Dado que desde sus inicios Android fue concebido como un proyecto *open-source*, se permitió y se permite una amplia revisión de seguridad por parte de cualquier persona interesada, lo que contribuye a la constante retroalimentación y a la mejora posterior de la plataforma.
- **Respuesta a incidentes:** Android creó un proceso que contempla dos aspectos para dar respuestas de seguridad. Primero, hay una vigilancia constante por parte del equipo de seguridad de Android de los componentes del sistema, y de la comunidad para identificar posibles vulnerabilidades. Segundo, una vez se descubren problemas, el equipo de seguridad tiene un proceso de respuesta que permite una rápida atención de las vulnerabilidades para que el riesgo potencial de los usuarios Android sea reducido al mínimo. Estas pueden incluir actualizaciones de la plataforma y la eliminación de aplicaciones de Google Play y de los dispositivos.

Uno de los objetivos de Android es ser el sistema operativo más seguro y útil para dispositivos móviles del mercado, proveyendo controles de seguridad periódicos que se enfocan en: proteger los datos de los usuarios, proteger los recursos del sistema (incluyendo la red) y proporcionar aislamiento de las aplicaciones. Para estos tres objetivos, Android proporciona características de seguridad claves (Méndez-Acuña and Sánchez, 2016, p5):

- **Seguridad a través del kernel de Linux:** A nivel de seguridad del sistema operativo, Android ofrece la seguridad del kernel Linux, el cual ha sido utilizado desde hace años y se usa en la seguridad de millones de ambientes sensibles[5]. El kernel de Linux ofrece características de seguridad como el que funciona con un modelo basado en permisos, contempla aislamiento de procesos, tiene un mecanismo de seguridad extensible y permite eliminar partes innecesarias y potencialmente inseguras.

[5] A. Alfonso. *Informe de seguridad en iOS y Android*, abril 2012.

- **Mecanismo de seguridad *sandbox* obligatorio para todas las aplicaciones:** En el kernel cada aplicación se ejecuta como un proceso con identificadores de grupo y usuario particulares. Lo que permite que las políticas de acceso puedan ser definidas para cada aplicación dependiendo de sus requerimientos y propósitos. Dichos permisos son aprobados por el usuario durante la instalación mediante la aceptación de términos y condiciones. En algunos SO errores de corrupción de memoria comprometen la seguridad completa del dispositivo, estos son minimizados en Android debido a que todas las aplicaciones y sus recursos están en espacios de ejecución aislados unos de otros, por lo que una corrupción en la memoria sólo se limitaría y afectaría a la aplicación a la cual se le haya asignado dicho espacio de memoria independiente.
- **Partición del sistema y modo seguro:** Cuando un usuario inicia el equipo en modo seguro, sólo se inicia la partición del sistema, por lo que sólo las aplicaciones básicas de Android estarán disponibles, lo que asegura al usuario un inicio de dispositivo en un ambiente libre de aplicaciones de terceros y, por lo tanto, en un modo seguro. La partición del sistema as aquella que contiene el kernel de Linux, las librerías, el tiempo de ejecución, el framework de aplicaciones y las aplicaciones. Esta es de sólo lectura.
- **Permisos del sistema de archivos:** Los permisos del sistema de archivos Linux aseguran que un usuario no puede leer ni escribir archivos de otro usuario. Dado que en Android las aplicaciones de ejecutan como su propio usuario, una aplicación no puede leer ni escribir archivos creados por otra app, a menos que el desarrollador de la misma los exponga de manera abierta.
- **Sistema de cifrado:** Desde la versión 3.0 de Android se proporciona cifrado completo de los ficheros. Todos los datos se cifran en el kernel utilizando una clave derivada de la contraseña del usuario para evitar ataques de fuerza bruta, ésta se combina con una sal al azar y un hash al momento de la

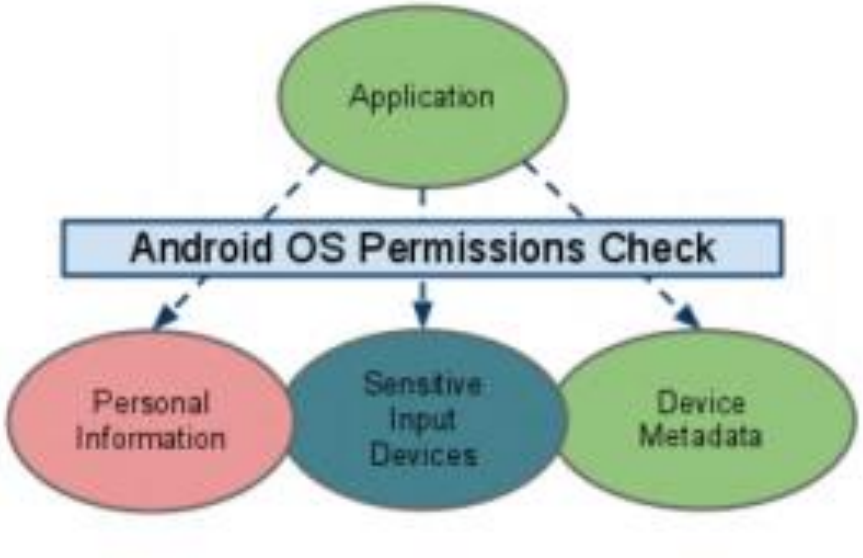
encriptación[6]. Para proporcionar resistencia contra ataques de diccionario, Android cuenta con reglas de complejidad de contraseñas que puede configurar el administrador del dispositivo y son ejecutadas por el sistema operativo.

- **Protección por contraseña:** Android se puede configurar para que solicite una contraseña de usuario antes de proporcionar acceso al dispositivo, esto facilita la prevención del uso no autorizado del dispositivo. Y, como se explicó anteriormente, la contraseña es utilizada por el algoritmo de encriptación.[7]
- **Administración de dispositivos:** Android proporciona un API para la administración de dispositivos que contienen funciones al nivel del sistema. Haciendo uso de esta API se puede hacer uso de funcionalidades como el borrado remoto de la información o restaurar valores de fábrica.
- **Mejora de seguridad en la administración de la memoria:** Android ha incluido características de seguridad que hacen difícil aprovechar los problemas comunes de corrupción de memoria. Algunas de estas características son, por ejemplo, escoger de forma aleatoria los lugares importantes de la memoria, reducir problemas de desbordamiento y evitar ejecución de código en la pila y el heap.
- **Permisos de "root" en los dispositivos:** En Android sólo el kernel y un subconjunto pequeño de aplicaciones principales tienen acceso y son ejecutadas con permisos raíz (root). Estos pueden modificar el sistema operativo, el núcleo y cualquier otra aplicación, además tiene acceso completo a los datos de aplicación.
  Android ha permitido que los permisos de root se puedan configurar, debido a que esta es una propiedad importante para los desarrolladores y para aquellos usuarios que, por ejemplo, quieren permitir la instalación de un SO alterno.
- **API protegidas:** Algunas API necesitan permisos especiales para poder ser utilizadas, estas API son: la de acceso a la información personal; la de uso

[6] H. Bing. *Analysis and research of system security based on Android.* En *Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference* en, páginas 581-584, 2012.
[7] A. Alfonso. *Informe de seguridad en iOS y Android*, abril 2012.

de dispositivos de entrada sensibles como la cámara, el micrófono o el GPS, y la de metadatos del dispositivo, que, aunque son datos poco sensibles por naturaleza, indirectamente pueden revelar características sobre el usuario, las preferencias del mismo y la forma en que utiliza el dispositivo[8].

Figura 2. API protegidas de Android.



Fuente: (http://source.android.com/tech/security/, s.f.)

- **Firma de las aplicaciones:** La solicitud de firma del código permite a los desarrolladores identificar el autor y responsable de la aplicación, las aplicaciones que intentan instalarse sin ser firmadas son rechazadas. Esta

[8] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, y C. Glezer. *Google Android: A comprehensive security assessment. Security Privacy, IEEE,* 8(2):35-44, 2010.

firma se hace por medio de certificados que son verificados al momento de la instalación.[9]

2.1.2 Antecedentes en seguridad en el control de accesos y seguridad de la información

El ministerio del interior establece las políticas y normas para garantizar un adecuado control de acceso a los sistemas de información del Ministerio del Interior y el Fondo para la Participación y el Fortalecimiento de la Democracia. Con base a esta normativa[10], se crea el trasfondo sobre el cual se erigen los antecedentes para la buena praxis sobre la seguridad de la información y el control de accesos para aplicaciones móviles.

2.1.2.1 Responsabilidades del Oficial de Seguridad de la Información

Sugerir procedimientos para la asignación de acceso a los sistemas, bases de datos y servicios de información multiusuario; la solicitud y aprobación de acceso a Internet o redes externas; el uso de computación móvil, trabajo remoto.

Analizar y sugerir medidas a ser implementadas para hacer efectivo el control de acceso de los usuarios a diferentes servicios como VPN, Internet o digitalización entre otros.
Verificar las pautas establecidas (su cumplimiento), relacionadas con control de acceso, creación de usuarios, administración de privilegios, administración de contraseñas, utilización de servicios de red, autenticación de usuarios, uso controlado de utilitarios del sistema.

2.1.2.2 Responsabilidades de los activos de información

[9] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, y C. Glezer. *Google Android: A comprehensive security assessment. Security Privacy, IEEE*, 8(2):35-44, 2010.
[10] Ministerio del Interior y el Fondo para la Participación y el Fortalecimiento de la Democracia. (2014). *Políticas para la Gestión de Seguridad de la Información – Control de Acceso*. Recuperado de: https://www.mininterior.gov.co/sites/default/files/oip-2014-psi-especificas-6_control_acceso.doc

Evaluar los riesgos a los cuales se expone la información con el objeto de:

- Clasificar la información.
- Determinar los controles de acceso, autenticación y utilización a ser implementados en cada caso.
- Aprobar y solicitar la asignación de privilegios a usuarios.
- Llevar a cabo un proceso formal y periódico de revisión de los Privilegios de acceso a la información.

2.1.2.3 Normas y Requerimientos para el Control de Acceso

Control Acceso:

Los controles de acceso deberán contemplar:

- Requerimientos de seguridad de cada una de las aplicaciones.
- Definir los perfiles o privilegios de acceso de los usuarios a las aplicaciones de acuerdo a su perfil de cargo en la entidad.

Administración de Acceso de Usuarios:

El administrador de la plataforma establece procedimientos para controlar la asignación de derechos de acceso a los sistemas, datos y servicios de información.

Creación de Usuarios:

El administrador de la plataforma, deberá mantener los registros donde cada uno de los lideres responsables de los procesos haya autorizado a los servidores públicos o terceros el acceso a los diferentes sistemas de información de la entidad.

Los datos de acceso a los sistemas de información deberán estar compuestos por un ID o nombre de usuario y contraseña que debe ser único por cada servidor público o tercero.

Cuando se retire o cambie de contrato cualquier servidor público o tercero, se deberá aplicar la eliminación o cambios de privilegios en los sistemas de información a los que el usuario estaba autorizado.

El proceso de gestión de información y comunicaciones deberá realizar revisiones de privilegios de acceso a los diferentes sistemas de información por parte de los servidores públicos y terceros, manteniendo los registros de las revisiones y hallazgos.

Administración de Contraseñas de Usuario:

Las contraseñas de acceso deberán cumplir con un mínimo de 8 caracteres y la combinación de números, letras mayúsculas y minúsculas, en lo posible utilizar caracteres especiales.

Todos los servidores públicos deberán cambiar su contraseña de acceso a los diferentes sistemas de información con una frecuencia mínima de 3 meses, a excepción de aquellos que contengan información confidencial o secreta en cuyo caso el cambio se debe realizar cada mes.

Los sistemas de información deberán bloquear permanentemente al usuario luego de 5 intentos fallidos de autenticación a excepción de aquellos que contengan información confidencial o secreta en cuyo caso después de 3 intentos fallidos de autenticación se realizará el bloqueo.

Uso de Contraseñas:

Los usuarios deben cumplir las siguientes normas:

- Mantener los datos de acceso en secreto.
- Contraseñas fáciles de recordar y difíciles de adivinar.

- Que las contraseñas no estén basadas en algún dato que otra persona pueda adivinar u obtener fácilmente mediante información relacionada con la persona, por ejemplo, nombres, números de teléfono, fecha de nacimiento, etc.
- Notificar de acuerdo a lo establecido cualquier incidente de seguridad relacionado con sus contraseñas: pérdida, robo o indicio de pérdida de confidencialidad.

Control de Identificación y Autenticación de Usuarios:

Todos los usuarios (incluido el personal de soporte técnico, como los operadores, administradores de red, programadores de sistemas y administradores de bases de datos) tendrán un identificador único (ID de Usuario) solamente para su uso personal exclusivo, de manera que las actividades tengan trazabilidad.

Sistema de Administración de Contraseñas:

El sistema de administración de contraseñas debe:

- Obligar el uso de UserID´s y contraseñas individuales para determinar responsabilidades.
- Permitir que los usuarios seleccionen y cambien sus propias contraseñas luego de cumplido el plazo mínimo de mantenimiento de las mismas o cuando consideren que la misma ha sido comprometida e incluir un procedimiento de confirmación para contemplar los errores de ingreso.
- Obligar a los usuarios a cambiar las contraseñas provisionales o que han sido asignadas por el administrador del sistema de información.
- No permitir mostrar las contraseñas en texto claro cuando son ingresadas.
- Almacenar las contraseñas en forma cifrada.

Sesiones Inactivas:

Si el usuario debe abandonar la estación de trabajo momentáneamente, activará protectores de pantalla con contraseñas, con el fin de evitar que Terceros puedan ver su trabajo o continuar con la sesión de usuario habilitada.

Si los sistemas de información detectan inactividad por un periodo igual o superior a diez minutos, deben automáticamente aplicar, “timeout” es decir, finalizar la sesión de usuario.

Limitación del Tiempo de Conexión:

Las restricciones al horario de conexión deben suministrar seguridad adicional a las aplicaciones de alto riesgo:

- Limitar los tiempos de conexión al horario normal de oficina, de no existir un requerimiento operativo de horas extras o extensión horaria.
- Documentar los funcionarios o contratistas que no tienen restricciones horarias y los motivos y evidencia de la autorización expedida por el líder del proceso de Gestión de Tecnologías de Comunicación e Información.

2.1.2.4 Consideraciones ante la normativa

La normativa anterior ha sido adaptada de las Políticas de Seguridad de la Información, Control de Acceso,[11] del Ministerio del Interior del gobierno colombiano, año 2014. Se han removido las políticas que no se consideran aplicables para el entorno preestablecido por el presente proyecto (Móvil: ANDROID), y que no entrarán en consideración para la elaboración del modelo 1.0 de la aplicación.

2.1.3 Antecedentes en seguridad de transacciones de banca móvil y la industria financiera[12]

[11] Ministerio del Interior y el Fondo para la Participación y el Fortalecimiento de la Democracia. (2014). *Políticas para la Gestión de Seguridad de la Información – Control de Acceso*. Recuperado de: https://www.mininterior.gov.co/sites/default/files/oip-2014-psi-especificas-6_control_acceso.doc

[12] *Information Security in Banking and Financial Industry. IJCEM International Journal of Computational Engineering & Management, vol. 14, October 2011.* Recuperado de: http://www.IJCEM.org. Fuente principal.

La historia de la banca en línea no es muy antigua, pero ha tenido importantes progresos durante dos décadas que han sido posibles debido a sus características propias. Ahora, la banca móvil y los movimientos online están al constante uso por parte de personas y empresas, al igual que centros de estudio y gobiernos por igual. En un estudio en Singapur, investigadores encontraron que las personas que aceptan el uso de la banca móvil u online comparados con aquellos que no, reconocieron que la consideran menos compleja, y mas compatible. Algunos investigadores argumentan que, aun así, la tecnología del *eBanking* todavía está falta de maduración.[13]

Sin embargo, para finales del 2002, algunas estimaciones aseguran que al menos el 30% de los estadounidenses realizaban sus operaciones bancarias de forma online, número que incrementó al 50% en 2003. De forma similar, otros estudios anticiparon que más de 20 millones de personas en el reino unido adoptarían el e-banking para fines del 2005. Esta tendencia aparentemente también se adoptó en Singapur, Suecia, Alemania y Noruega, y las más avanzadas y autosuficientes economías del mundo, también en la India. Sin embargo, la seguridad jugó un papel muy importante en adoptar la banca en línea y la continuación de su uso que indica que la seguridad es un tema directamente enlazado con la adopción de la banca online. (Ambhire, Teltumde 2011)

2.1.3.1 Vista técnica

La seguridad de la información significa la protección tanto de los datos como de los sistemas de información de accesos no autorizados, uso, publicación, modificación, interrupción, investigación, inspección, copiado, grabado o destrucción. Los elementos son confidencialidad, posesión, integridad, autenticidad, disponibilidad y utilidad.

[13] Online Banking Report. The Online Banking Report, 1999. Recuperado de http://www.onlinebakingreport.com.

Confidencialidad es el término usado para prevenir la publicación de información no autorizada sobre individuos o sistemas. Por ejemplo, una transacción de tarjeta de crédito en internet requiere que el número de la misma sea transmitido desde el comprador al vendedor y del vendedor a una red de procesos transaccionales. El sistema intenta asegurar la confidencialidad encriptando el número de la tarjeta durante la transmisión, limitando los lugares donde pueda aparecer (en bases de datos, registros, *backups,* recibos impresos, etc.), y restringiendo los lugares donde es guardado. Si un grupo no autorizado obtiene el número de la tarjeta de crédito de cualquier forma, ha tenido lugar una violación a la confidencialidad. La confidencialidad es necesaria (pero no suficiente) para mantener la privacidad de las personas cuya información es suministrada a un sistema de información.

Integridad significa que la información no puede ser modificada de forma indetectable. Esto no es lo mismo que integridad referencial en bases de datos, aunque también puede ser visto como un caso especial de Consistencia como es entendido en el modelo clásico de ACID[14] en los procesos transaccionales. La integridad es violada cuando un mensaje es activamente modificado en tránsito. (Ambhire, Teltumde 2011).

Para cualquier sistema de información que sirva su propósito, la información debe estar disponible cuando se necesite. Esto significa que el sistema usado para guardar y procesar la información, los controles de seguridad usados para protegerlo, y los canales de comunicación usados para accederlo deben estar funcionando correctamente.
Los sistemas de alta disposición apuntan a mantener su disponibilidad en todo momento, previniéndose ante interrupciones de servicio debido a fallas de energía, fallas de hardware, y actualizaciones de sistema.

[14] *Principles of transaction-oriented database recovery*. ACM Computing Surveys. Haerder, Reuter. 15(4): Punto 287.

Asegurar la disponibilidad también significa prevenir ataques de denegación de servicio (comúnmente DDOS). No-repudio implica que la intención de uno es cumplir sus obligaciones por contrato. También implica que uno, como parte de un agente en una transacción, no puede negar haber recibido la misma ni la otra parte puede negar haberla enviado. (Piedad, Hawkins 2011).

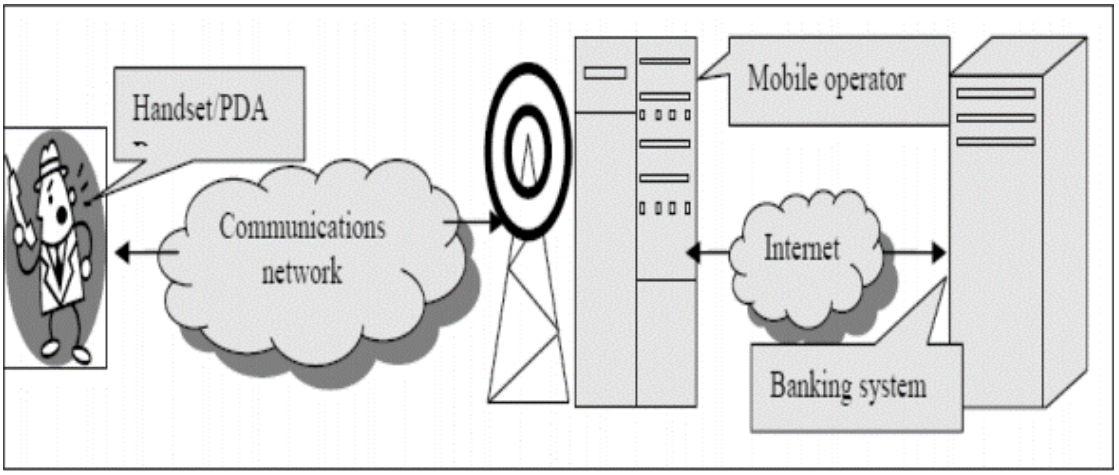
2.1.3.2 Vista bancaria

Con la expansión abrumadora de las telecomunicaciones móviles y la tecnología en el mundo de los negocios, la banca móvil se convirtió en un método de manejo bancario popular y prometedor en la industria. La banca móvil puede proveer a los clientes con una mejor calidad de servicios a menores costos. Se refiere a la provisión y disposición de servicios bancarios y financieros con la ayuda de los dispositivos de telecomunicaciones móviles. El alcance de los servicios promovidos puede incluir facilidades para conducir transacciones de banca e inversión en el mercado, la administración de cuentas y el acceso a información personalizada. La mayoría de los investigadores bancarios concuerdan en que la banca móvil consiste en tres partes: contabilidad móvil, carretaje móvil y servicios de información financiera móvil. Los servicios para clientes incluyen: revisión del balance, transacciones de cuentas, pagos, etc. Servicios de un banco convencional. De manera cada vez mayor, los clientes bancarios esperan información en tiempo real y acceso las veinticuatro horas del día, siete días a la semana, en cualquier lugar del mundo. Servicios como manejo de cuentas electrónicas, carretaje móvil e información financiera y alertas, aseguran a los operadores de las redes bancarias una ventaja competitiva sobre la fidelidad de los clientes. (Ambhire, Teltumde 2011).

Un sistema de banca móvil comprime una unidad bancaria y el centro de procesado que puede ser el computador *mainframe* del banco responsable de las transacciones y el manejo de la información. La banca móvil incluye una o más terminales bancarias como cajeros, maquinas de deposo y estaciones de información multimedia.

Los sistemas de banca móvil han provisto una buena base para proveer nuevos modelos de servicios personalizados, orientados al cliente, que incorporan una gran cantidad de canales de comunicación móvil, integrando grandes méritos desarrollados por diferentes ramas tecnológicas. (Ambhire, Teltumde 2011).

Figura 3. Sistema de operación de banca móvil.



Fuente: (semanticscholar.org/3475/8c819d3bb4d4e5e4b4d053db523684adac09.pdf, s.f.)

2.1.3.3 Vista financiera

Las tecnologías más importantes incluyen: Criptografía, esteganografía, PKI (Public Key Infrastructure), firmas electrónicas, certificados electrónicos, protocolos de seguridad, protocolos de autenticación, firewalls y proxis, modelos de control de accesos, contraseñas, sobres digitales, tecnologías de seguridad biológica, filtrado, sistemas de detección de intrusiones. (Ambhire, Teltumde 2011).

2.1.4 Vulnerabilidades en aplicaciones

El tremendo incremento en las transacciones online ha sido acompañado por un alza similar en el número y tipos de ataques en contra de los sistemas de seguridad. Algunos de esos ataques han utilizado vulnerabilidades que han sido publicadas en componentes de terceros utilizados por las aplicaciones, como software de carritos de compra. Otros ataques han utilizado vulnerabilidades que son comunes para

cualquier aplicación como *SQL Injection* o *cross-site scripting.* La exploración exitosa de estas vulnerabilidades puede llevar a la prevención de ataques y la mejora de la seguridad. (Top 10 Vulnerabilities in web applications. OWASP, 2016-actualizado 2018)

1. **SQL Injection:** Se refiere a la inserción de meta-caracteres SQL desde el punto del usuario, de tal forma que las instrucciones del atacante son ejecutadas en el back-end del sistema. Usualmente, los atacantes determinarán primero si el sitio es vulnerable a este tipo de ataques enviando un carácter sencillo (como una comilla “”). El resultado de una inyección de SQL en un sistema vulnerable puede variar desde un error de mensaje detallado, que revela la tecnología usada en el back-end, o que permite que el atacante acceda a áreas restringidas del sitio o la aplicación porque manipuló alguna orden cambiando un valor booleano a un *always-true*, o que incluso se permita la ejecución de comandos asociados al sistema operativo.
2. **Broken Authentication:** Esta ocurre cuando la aplicación maneja de forma indebida las sesiones relacionadas el usuario, de tal forma que la información del mismo se ve comprometida. La información puede ser almacenada en formularios se sesión de *cookies*, contraseñas, claves secretas, *keys*, etc. El objetivo de este ataque es entrar en la sesión de otra persona o usar una sesión que ha sido terminada por el usuario o robar información relacionada con la sesión.
3. **Exposición de información sensible:** Si la información no es manejadad e forma debida por la aplicación, un atacante puede robar o modificar información sensible. Unos ejemplos incluyen el uso de claves de encriptación débiles, o TLS débiles.
4. **XML External Entities (XXE):** Una aplicación es vulnerable a un ataque XXE si esta acepta que los usuarios suban archivos XML maliciosos que puedan explotar una codificación vulnerable o ciertas

dependencias. Esto se puede utilizar para ejecutar código, robar información y realizar otras tareas maliciosas.

5. **Control de accesos rotos:** Este ocurre si al usuario le es posible acceder a recursos no autorizados, que pueden ser el acceso a paginas restringidas, bases de datos, directorios, etc. Las aplicaciones que manejan varios tipos de cuentas dependiendo del usuario: administrador, operador, grupos de reportes, moderador, etc. Un problema común es que los desarrolladores restringen los privilegios sólo en la vista del usuario (interfaz de usuario) y no del lado del servidor. Si esta vulnerabilidad se explota se pueden asignar privilegios de administrador a un usuario común.
6. **Mala configuración de seguridad:** Los desarrolladores y el equipo de IT se aseguran de la funcionalidad mas no de la seguridad. Las configuraciones son realizadas desde el servidor de la aplicación, el servidor de base de datos, proxy, aplicaciones y otros dispositivos necesitan estar alineados con los requerimientos de la política de seguridad. La mayoría de los requerimientos de seguridad son pasados por alto a no ser que sean identificados por un experto o un hacker. Ejemplos de esto son la configuración de contraseñas débiles, contraseñas por defecto, scripts por defecto almacenados en el servidor, directorios por defecto, mensajes de error por defecto, etc.
7. **Cross Site Scription (XSS):** El *cross-site scripting* ocurre cuando a un atacante le es posible insertar data o scripts de poca confianza en una página web. La data/script insertado por el atacante es ejecutada por el buscador que puede entonces robar la información del usuario, desfasar un sitio web, etc. Existen tres tipos *Reflected, Stored, DOM-based*.
8. **Deserialización insegura:** Algunas aplicaciones guardan información en el lado del cliente y pueden estar usando serialización de objeto (object serialization). Las aplicaciones que dependen en el cliente para

mantener un estado pueden permitir la manipulación de información serializada.

9. **Usar componentes con vulnerabilidades conocidas:** Si algún componente utilizado tiene vulnerabilidades conocidas este puede llevar a brechas de seguridad o una toma total del servicio. Estos componentes pueden ser frameworks, librerías, funciones vulnerables, frameworks de red, etc.
10. **Monitoreo insuficiente:** Con todas las contramedidas en su lugar los ataques aún pueden suceder y aun así ser detectados una vez el ataque ya ha sucedido. Si no es detectado, los atacantes pueden haber dejado el sistema comprometido para futuras intrusiones, obteniendo de esta forma persistencia. Para asegurar que el intento malicioso por parte de un atacante sea encontrado de antemano, es esencial que toda la actividad de la aplicación sea monitoreada en busca de cualquier comportamiento sospechoso.

En contraposición a las vulnerabilidades listadas por OWASP orientadas de manera específica a aplicaciones móviles. (A Conceptual Exploration for the Safe Development of Mobile Devices Software Based on OWASP. (s.f.) Fundación Universitaria los Libertadores, Bogotá D.C., Colombia. Gil, Baquero, Plazas, Hernández. 2018.)

1. **Uso inapropiado de la plataforma:** Esta categoría cubre el uso inapropiado de la plataforma o de la falta de uso de los controles de seguridad, permisos de la plataforma, manipulación indebida de TouchID, servicios de contraseñas (KeyChain IOS) o algún otro sistema de control de seguridad que es parte del sistema operativo móvil.
2. **Almacenamiento no-seguro de información:** Esta nueva categoría abarca todo lo relacionado con el almacenamiento indebido de la información y filtraciones no deseadas de información.

3. **Comunicación no-segura:** Esta cubre los protocolos de enlace pobres, versiones incorrectas del SSL, negociaciones débiles, comunicación desencriptada de información sensible, etc.
4. **Autenticación insegura:** Esta categoría incluye las nociones de autenticación del usuario final o manejo incorrecto de sesiones. Esto incluye: No identificar al usuario del todo cuando es necesario, fallar en mantener la identidad del usuario cuando es requerido y un manejo débil de sesiones.
5. **Criptografía insuficiente:** El código aplica criptografía a los bienes que se consideran información sensible. Sin embargo, la misma en insuficiente en algunos casos.
6. **Autorización insegura:** Esta captura cualquier fallo no autorizado (por ejemplo, autorización de decisiones del lado del cliente, navegación forzada, etc.). Es diferente a los problemas de autenticación.
7. **Calidad del código del cliente:** Este contiene todos los problemas asociados a la implementación indebida del código al nivel del cliente móvil. Esta es diferente de los errores de codificación del servidor. Esta contiene casos como *buffer overflow*, vulnerabilidades del formato de cadenas, y varios otros errores a nivel de codificación cuya solución es reescribir el código que está corriendo en el dispositivo móvil.
8. **Adulteración de código:** Esta categoría cubre los parches binarios, modificaciones de recursos locales, ganchos de métodos, modificación dinámica de la memoria y enredo de métodos. Una vez la aplicación es entregada al dispositivo móvil, el código y los recursos de información residen allí. Un atacante que modifica directamente el código, cambia de forma dinámica el contenido de la memoria, cambian o remplaza las APIs del sistema que son usadas por la aplicación o modifica los recursos y la información de la misma.

9. **Ingeniería inversa:** Esta categoría incluye el análisis de kernel binario final para determinar su código fuente, librerías, algoritmos y otros bienes. El software como IDA pro, Hopper, oTool, y otros inspectores binarios dan al atacante la visualización de la operación interna de la aplicación, así como revelar información sobre el servidor back-end, constantes criptográficas y figuras, y la propiedad intelectual.
10. **Funcionabilidad extraña:** Muchas veces, los desarrolladores incluyen funcionalidades ocultas de puerta trasera u otros controles de seguridad internos que no están hechos para ser puestos en marcha en un ambiente de producción. Por ejemplo, un desarrollador puede accidentalmente incluir una contraseña como un comentario en una aplicación hibrida.

2.1.5 Política de seguridad.

Una política de seguridad es un documento que define como encara la empresa la seguridad de sus recursos IT. El alcance aquí puede ser muy amplio: seguridad física, seguridad de red, seguridad de la información y (especialmente importante en este caso) seguridad de dispositivos móviles. Es muy importante que una política de seguridad sea puesta en marcha antes de que cualquier decisión tomada como solución sea hecha. Mientras que las medidas de seguridad básicas deben estar siempre operacionales, muy a menudo grandes y costosas actualizaciones son hechas antes de que la necesidad de estas sea establecida. Una política de seguridad, por lo tanto, sirve como un punto de enfoque vital para cualquier empresa u organización. (Mathias, Craigh. 2008).

En general, una política de seguridad define qué información será tratada como sensible (y por lo tanto protegida), quien debería tener acceso a esta información y bajo qué condiciones, y -muy importante- qué hacer cuando la seguridad es comprometida, o bajo sospecha de estar comprometida. Desde un punto de vista

físico, debería definir quien tiene acceso a ciertas áreas de una instalación o edificio, y cómo serán protegidas dichas áreas. Pero los siguientes tres elementos son más importantes respecto a la seguridad móvil:

- **Seguridad de la información:** Define qué información deberá ser tratada como sensible. Generalmente se recomienda adoptar una perspectiva similar a la utilizada por el gobierno, la cual es tener múltiples niveles de seguridad con menor personas o grupos con acceso a la par que los niveles de clasificación aumentan. Por ejemplo, “La compañía utilizará una leyenda de ‘Company Confidential’ para cualquier información restringida bajo acceso de sólo empleados, y ‘Company Most Confidential’ para aquel bajo acceso de sólo alguno empleados. Esta política aplica también para restringir acceso a ciertas aplicaciones para sólo usuarios.”
- **Seguridad de red:** Existen dos elementos clave a solucionar aquí, una autenticación y encriptación fuertes de la información sensible sin importar donde es almacenada. Tomar en cuenta que mientras una política de seguridad usualmente no ha de definir una solución puntal para todos y cada uno de los posibles casos, debe mencionar, por ejemplo, que una autenticación mutua es requerida a la par que un VPN debe ser utilizado cuando se acceso a la red de la empresa de manera remota. Incluso podría restringir una red especifica para ciertos portadores aprobados.
- **Seguridad de dispositivos:** Esta parte de la política define cómo se mantendrá la seguridad en los dispositivos móviles fuera del perímetro y la protección de la empresa física (incluyendo aquí la seguridad para tecnologías como BYOD). Esta puede restringir la capacidad de un usuario para instalar aplicaciones en el dispositivo, por ejemplo, y podría especificar que el dispositivo ha de ser respaldado o chequeado por virus, o que un firewall en particular sea configurado en el mismo.

2.1.6 Legales

Normatividad sobre derechos de autor y propiedad intelectual en Colombia.

Constitución Política de Colombia.

Artículo 61. El estado protegerá la propiedad intelectual por el tiempo y mediante las formalidades que establezca la ley. Nota: El concepto de “propiedad intelectual”, acogido por el artículo 61 de la Constitución Política, en concordancia con el artículo 2 numeral 8 del Convenio que establece la Organización Mundial de la Propiedad Intelectual, es omnicomprensivo de diferentes categorías de propiedad sobre creaciones del intelecto, que incluye dos grandes especies o ramas: la propiedad industrial y el derecho de autor, que, aunque comparten su naturaleza especial o sui generis, se ocupan de materias distintas. Mientras que la primera trata principalmente de la protección de las invenciones, las marcas, los dibujos o modelos industriales, y la represión de la competencia desleal, el derecho de autor recae sobre obras literarias, artísticas, musicales, emisiones de radiodifusión, programas de ordenador, etc.

Ley 23 de 1982 Sobre los Derechos de Autor

2.1 Articulo 1.

Los Autores de obras literarias, científicas y artísticas gozarán de protección para sus obras en la forma prescrita por la presente ley y, en cuanto fuere compatible con ella, por el derecho común. También protege esta ley a los intérpretes o ejecutantes, a los productores de fonogramas y a los organismos de radiodifusión, en sus derechos conexos a los del autor.

2.2 Articulo 2.

Los derechos de autor recaen sobre las obras científicas, literarias y artísticas las cuales se comprenden todas las creaciones del espíritu en el campo científico, literario y artístico, cualquiera que sea el modo o forma de expresión y cualquiera que sea su destinación, tales como: Los libros, folletos y otros escritos (…)

(http://www.cide.edu.co/cidevirtual/file.php/1/Normatividad_Derechos_de_autor.pdf , s.f.)

2.2 BASES TEORICAS

2.2.1 Lenguajes de programación

Los lenguajes de programación de una computadora en particular se conocen como código de máquinas o lenguaje de máquinas. A diario se interactúa con distintos tipos de máquinas como teléfonos móviles, tabletas y computadoras. Todos ellos tienen uno o varios lenguajes de programación que le ayudan a traducir las ordenes del usuario en comandos interpretables por la unidad, o la red.

Un lenguaje de programación es básicamente un sistema estructurado de comunicación, similar al humano, el cual nos permite comunicarnos por medio de símbolos y señales, ya sean palabras, sonidos o gestos. Refiriéndonos a los aparatos, este sistema está organizado para que se entiendan entre sí y a su vez interpreten las instrucciones a ejecutar, ya se entre usuario y máquina, o entre máquinas. (Colombiadigital, 2016).

2.2.2 Bases de datos

Conjunto de informaciones que está organizado y estructurado de un modo específico para que su contenido pueda ser tratado y analizado de manera rápida y sencilla.

Las bases de datos, por lo tanto, presentan datos estructurados de acuerdo a diferentes parámetros. Al disponer la información de una cierta forma, el usuario puede encontrar aquello que busca con facilidad, a diferencia de lo que le sucedería

si todos los datos estuvieran mezclados y sin ningún tipo de orden. (https//definición.de/base-de-datos/, s.f.)

MySQL es la base de datos de código abierto más popular del mundo. Con su rendimiento, confiabilidad y facilidad de uso comprobados, MySQL se ha convertido en la principal opción de base de datos para aplicaciones basadas en la web, utilizada por propiedades web de alto perfil como Facebook, Twitter, YouTube, y los cinco principales sitios web. Además, es una alternativa extremadamente popular como base de datos integrada, distribuida por miles de ISV y OEM. (https://www.oracle.com/co/mysql/index.html, s.f.)

2.2.3 Interfaz de usuario (UI)

La UI, en el campo de diseño industrial orientado a la interacción humano-maquina, es el espacio donde las interacciones entre humanos y máquinas ocurre. El objetivo de esta es permitir la operación efectiva y el control de la máquina desde el lado del humano, mientras que la máquina simultáneamente provee información que ayuda al proceso de la toma de decisiones por parte del operador[15].

Generalmente, el objetivo del diseño de la interfaz gráfica es producir un espacio que haga fácil, eficiente y disfrutable (*user-friendly)* el operar una máquina (o aplicación) de forma que esta arroje el resultado deseado. (Griffin, Baston. 2014)

2.2.4 Testing

El *testing* de software es una investigación conducida con el objetivo de proveer a los diseñadores con información acerca de la calidad del producto de software o servicio bajo prueba. (Kaner, 2006)

[15] *Interfaces* (presentación). *The user interface of mechanical system, a vehicle or an industrial installation is sometimes referred to as the human-machine interface (HMI)*. Ben Griffin, Laurel Baston. 2014.

El testeo de software también puede proveer una objetiva e independiente visión del software, lo que permite al negocio apreciar y entender los riesgos al momento de la implementación del software. Las técnicas de testing incluyen el proceso de ejecutar el programa o aplicación con la intención de encontrar *bugs* (errores u otros defectos), y verificar que el producto de software esta listo para su uso. (Pan, 1999)

Las pruebas también incluyen la ejecución de un componente del software o un componente del sistema para evaluar una o mas propiedades de interés. En general, estas propiedades indican la extensión ante la cual se someterá el sistema a prueba:

- Alcanza los requerimientos que guiaron su diseño y desarrollo.
- Responde correctamente a toda clase de estímulos de entrada.
- Realiza sus funciones dentro de un margen de tiempo aceptable.
- Es lo suficientemente usable.
- Puede ser instalado y corrido en todos los entornos considerados.
- Cumple con los resultados generales esperados por los diseñadores.

- **Prueba de caja blanca.**

  La prueba de caja blanca (también conocida como prueba de caja clara, prueba de caja de cristal, prueba de caja transparente, y prueba estructural) verifica la estructura interna de un programa, de forma contraria a la exposición funcional final presentada al usuario. En las pruebas de caja blanca, una prespectiva interna del sistema (el código fuente), así como las técnicas de programación, son usadas para diseñar los casos de prueba. El *tester*[16] elige la información o los pasos a ingresar para navegar internamente a través del código y determinar los resultados apropiados. Este testeo es análogo a la prueba de nodos en un circuito (Limaye, 2009).
- **Prueba de caja negra.**

[16] Persona encargada de llevar a cabo las pruebas, pudiendo ser parte del equipo de programación o diseño. O un tercero ajeno al proyecto.

La prueba de caja negra (también conocida como prueba funcional) trata el sistema como una “caja negra”, examinando su funcionalidad sin el conocimiento de la implementación interna, sin tener el código fuente presente. El *tester* sólo está consciente de lo que se supone debería hacer el software, no el cómo lo hace. Los métodos de prueba de caja negra incluyen: particionado equivalente, análisis de valores límites, *all-pairs testing*, tablas de estados de transición, tabla de pruebas de decisión, prueba aleatoria, prueba basada en el modelo, pruebas de casos de uso, pruebas exploratorias, y pruebas específicas del diseño. (Limaye, 2009)

2.2.5 Lenguaje unificado de modelado (UML).

UML, por sus siglas en inglés de *Unified Modeling Language*. Las grandes aplicaciones empresariales, los que ejecutan aplicaciones de negocios, deben ser algo más que un grupo de módulos de código. Ellos deben estructurarse de una manera que permite la escalabilidad, seguridad, y la ejecución bajo condiciones de estrés. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguaje de programación, esquemas de bases de datos y compuestos reciclados. Los mismos se dividen en dos clases[17]: Estructurales y de comportamiento[18]. (Fowler, Sccott. 1999)

- **Estructurales:**
    - Diagrama de clases.
    - Diagrama de componentes.
    - Diagrama de despliegue.
    - Diagrama de objetos.

[17] *Utilización de UML en Ingeniería del Software con Objetos y Componentes.* Stevens, Pooley, Wesley. 2002.
[18] No todos los diagramas serán aplicados en el presente proyecto.

    - Diagrama de paquetes.
    - Diagrama de perfiles.
    - Diagrama de estructura compuesta.
- **De comportamiento:**
    - Diagrama de actividades.
    - Diagramas de casos de uso.
    - Diagrama de máquina de estados.
    - Diagrama de interacción:
        - Diagrama de secuencia.
        - Diagrama de comunicación.
        - Diagrama de comunicación.
        - Diagrama de tiempos.
        - Diagrama global de interacciones.

2.2.6 Análisis y diseño orientado a objetos

La metodología de análisis y diseño orientado a objetos se ha usado ampliamente en el desarrollo de aplicativos orientados a la simulación, y al mismo tiempo se ha convertido en la metodología estándar en la industria del software, considerada también como una de las mejores prácticas para desarrollar proyectos de software con calidad.

Debido a su sencillez, esta metodología abarca de manera muy general la estructura de las interfaces de software haciendo énfasis solamente en las entradas y salidas de cada módulo, sin entrar en detalles de cómo se almacenan las variables o estructuras de datos en cada procedimiento. (Pressman, 2001).

2.2.7 Metodología tradicional

En su nacimiento (1968), la gestión de proyectos de software intentó imitar la gestión de proyectos de otras disciplinas, como la arquitectura, la industria o la ingeniería civil. Hereda y adapta al mundo del software muchos de sus roles (arquitectos de software) y tipos de organizaciones (fábricas de software). Por lo tanto, la manera de gestionar los proyectos es una conclusión lógica de la elección de imitar a la construcción (Gabino Distro, 2017):

- Modelo dirigido por la planificación, desarrollo tradicional, en cascada o predictibilidad.
- Se divide el proyecto en fases.
- No se pasa de una fase a otra hasta que no se acabe la anterior.
- Se especializa cada fase a un rol distinto: toma de requisitos, analistas, diseñadores, arquitectos técnicos, programadores, tester, etc.
- No se puede volver hacia atrás.

2.2.8 Metodología ágil

La principal diferencia entre los modelos tradicionales y los ciclos de vidas ágiles es que en estos últimos se asume que el análisis, diseño, etc., de cada iteración o Sprint son impredecibles. Los Sprints, o iteraciones cortas, no son (a priori no tienen porqué) lineales y son flexibles.

Finalmente, cada metodología de las llamadas ágiles, FDD, Crystal, Kanban, XP, Scrum, etc., matizará su ciclo de vida (Gabino Distro, 2017):

- Entregar prototipos, pequeños evolutivos que los usuarios puedan tocar.
- Con funcionalidad añadida en cada iteración.
- Pero también refinado y refactorizado con base en el *feedback* de los usuarios.
- Ciclo de vida iterativo e incremental.

- Iteraciones cortas en tiempo, de pocas semanas.

Sus ventajas teóricas son:

- Alcance totalmente abierto. Por lo tanto, gran capacidad de adaptación al cambio.
- Planificación de sprints cortos con prototipos en producción. Por lo tanto, el coste se puede asociar a cada sprint.
- Colaboración estrecha e intensa con el cliente (producto owner, retrospectivas).
- Producto completamente adaptado a requisitos de negocio en la entrega.

2.2.9 Metodología ágil SCRUM

SCRUM es un proceso de la metodología ágil que se usa para minimizar los riesgos durante la realización de un proyecto, pero de manera colaborativa.

Entre las ventajas se encuentran la productividad, calidad, y que se realiza un seguimiento diario de los avances del proyecto, logrando que los integrantes estén unidos, comunicados y que el cliente vaya viendo los avances.

En primer lugar, se define el *Product Backlog*, lo que nos permitirá realizar los *Sprint* del proyecto. (Scrum en la Metodología Ágil, 2017)

- **Product Backlog[19]**

  Es una lista sobre las funcionalidades deseadas del producto. Es elaborado por el dueño del producto y las funciones están priorizadas según lo que es más y menos importante para el negocio. El objetivo es que el dueño responda a la pregunta "¿Qué hay que hacer?". Definimos los objetivos y los alcances del producto que se va a desarrollar priorizando las necesidades.

- **Sprint Blacklog**

  Es un subconjunto de ítems del *Product Backlog*, que son seleccionados por el equipo para realizar durante el Sprint sobre el que se va a trabajar. El equipo establece la duración de cada Sprint. Los avances de cada una de las fases presentados al docente, en donde se sugieren los cambios pertinentes.

- **Sprint planning meeting**

  Es una reunión que se hace al comienzo de cada Sprint y se define cómo se va a enfocar el proyecto que viene del Product Backlog y las etapas y los plazos. Cada Sprint está compuesto por diferentes características a enfocar. Se define en el cronograma de actividades.

- **Scrum o stand-up meeting.**

  Es una reunión breve que se realiza a diario mientras dura el periodo de Sprint. Se responden individualmente tres preguntas: ¿Qué hice ayer?, ¿Qué voy a hacer hoy? Y ¿Qué ayuda necesito?

- **Sprint review.**

  Se revisa el Sprint una vez completado, y ya debería haber pruebas de un avance claro y tangible para presentar al cliente. Reuniones periódicas que nos garantizan la verificación de los avances y la compleción de las tareas a realizar en el pasado Sprint.

2.3 METAS A ALCANZAR

[19] Adaptado de: (http://www.i2btech.com/blog-i2b/tech-deployment/para-que-sirve-el-scrum-en-la-metodologia-agil/, s.f.*)*

Implementar la versión 1.0 del sistema denominado S.A.S (Sistema de Análisis de Seguridad) para abreviar, que permita dar cuenta de todo el proceso de ingeniería aplicado a un proyecto tangible de desarrollo de software para plataformas móviles. Aplicado en beneficio de la sociedad, empresas o personas que quieran asegurar el bienestar de la información y las transacciones seguras en sus plataformas.

2.3.1 A corto plazo

Realizar el levantamiento preliminar de la información actual sobre el estado de los sistemas de seguridad para plataformas móviles, y en seguridad de la información. La delimitación del alcance, definición de requerimientos funcionales y no funcionales, y la definición de los objetivos alcanzables.

2.3.2 A mediano plazo

Realizar el análisis de la información y las conclusiones preliminares realizadas con anterioridad sobre los alcances y las metas. Crear los cimientos del back-end y front-end de la aplicación de software. Y el desarrollo de la base de información.

2.3.3 A largo plazo

Concluir la realización de los entregable del producto en su versión prototipo 1.0, teniendo en cuenta la calidad y el cumplimiento de los objetivos propuestos.

2.4 PRODUCTOS A ENTREGAR

- **Manual técnico del desarrollo del software S.A.S en su versión 1.0:** Guía con conocimientos técnicos y anotaciones sobre la creación y programación del software. Lenguajes utilizados, modelos, información sobre la composición de los módulos y las reglas de iteración, etc.

- **Guía de usuario para la versión 1.0:** Producto base que provee la toma de contacto entre el software en su versión inicial y el usuario, explica paso a paso el uso desde el inicio del aplicativo hasta el guardado de la información provista por el mismo en materia de seguridad.
- **Software, versión 1.0:** Versión inicial del aplicativo que analiza, itera y arroja resultados acordes a la personalización del usuario, sugiriendo el sistema (o sistemas) de seguridad que mejor se acomoden a sus necesidades, tomando en cuenta tanto costos de tiempo y dinero, así como las reglas del negocio.

3. DISEÑO METODOLÓGICO

3.1 FASES DEL PROYECTO

La metodología propuesta se plantea en las siguientes etapas o fases del proyecto que se describen a continuación:

Fase de planeación.

La primera fase del proyecto, durante la cual se evaluaron los tiempos, tareas, actividades y se generó una lista de los requerimientos y componentes necesarios para la ejecución de las distintas fases del proyecto, verificando los costos estimados en tiempo y dinero de los distintos recursos.

Fase de revisión bibliográfica.

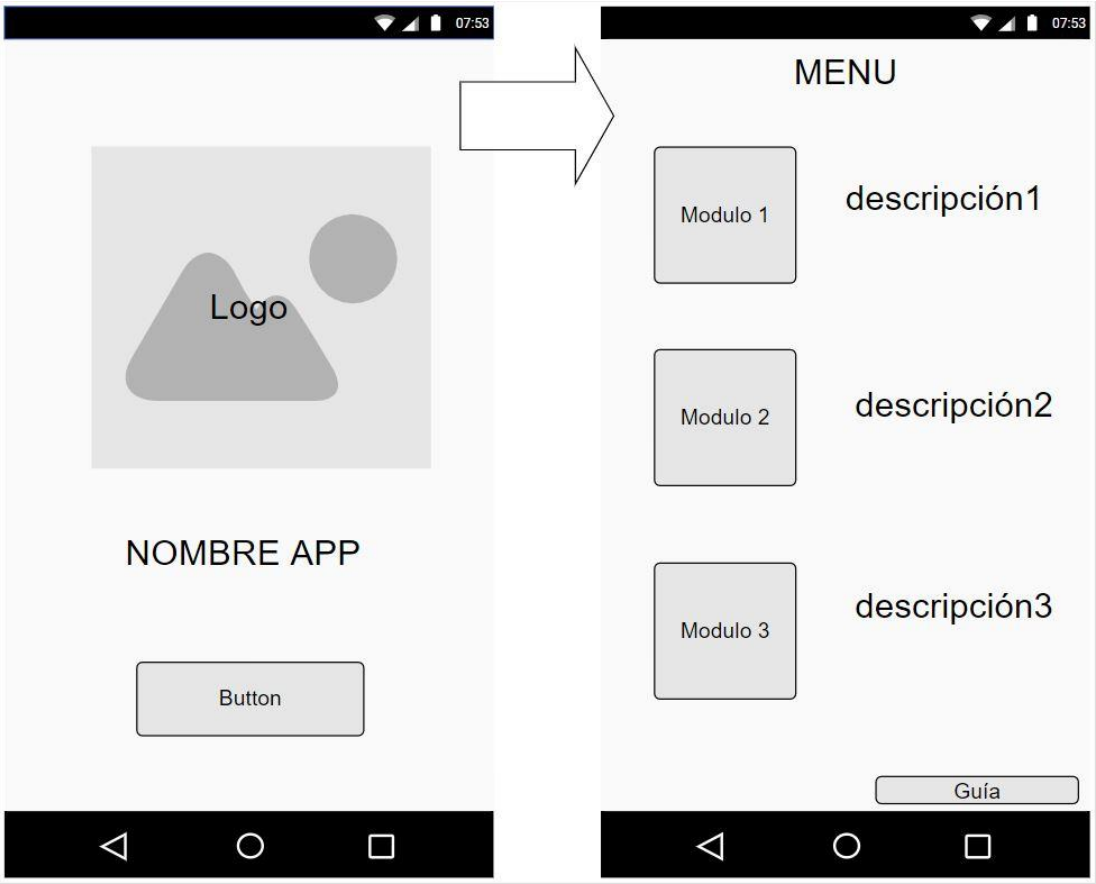
Durante esta etapa se realizó una consulta a las distintas bases de datos provistas por la universidad, y unas bases de dato externa provista por un patrocinador, adicionalmente se consultaron libros, páginas web, videos y otra documentación que contenía la información relacionada con las aplicaciones móviles, la seguridad de la información y la seguridad bancaria; los considerados tres temas principales a investigar.

Fase de ejecución.

En la fase de ejecución se realizó todo el proceso de programación y diseño gráfico, y para esto se hizo uso de dos herramientas principalmente: Photoshop para el diseño de la interfaz y el mapeado superficial, y Android Studio para la programación en ANDROID y el desarrollo FrontEnd y BackEnd. Inicialmente se realizaron varias pruebas que utilizaban distintos menús de estilo secuencial para ingresar a los módulos, pero se terminó descartando esta idea por un acercamiento más global, con un menú que contuviese todos los accesos al mismo tiempo. Aquí mismo se decidió optar por una paleta de colores metálica, basada en colores azulados y fríos para contener la información, generando así que al abrir el aplicativo no se

encontrara saturado con colores de todo el espectro, sino manteniendo una imagen calmada y directa, sin ningún color que desentone con la intención de guiar la atención del usuario a ningún otro sitio que no sea la aplicación y sus módulos.

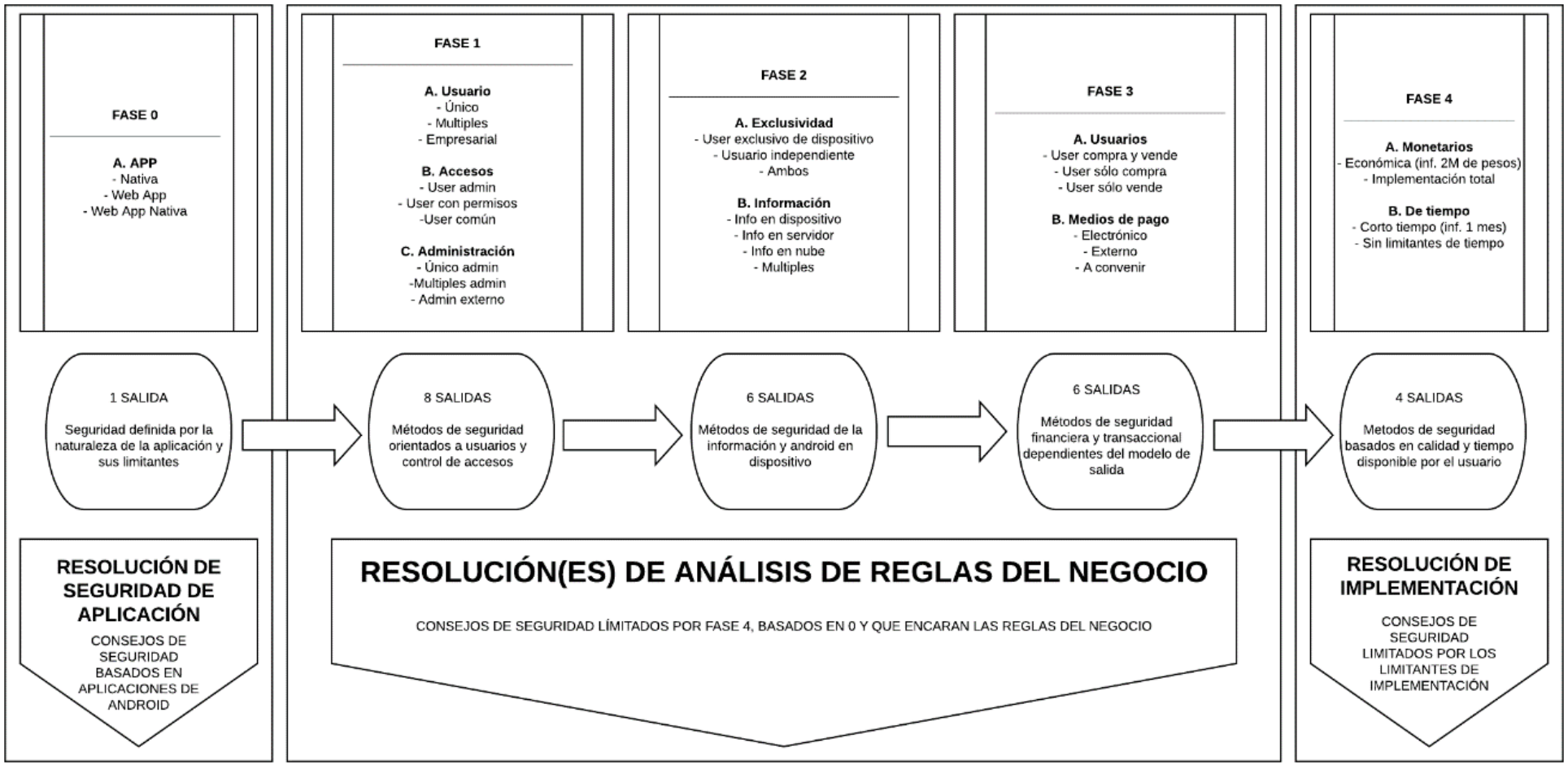
Figura 4. Mockup del inicio y el menú principal en su versión más temprana.



Una vez encarada la parte gráfica, se afrontó la parte que se consideró la más importante: el diseño de las reglas y el mapeado de las secuencias que llevarían al módulo de análisis de recibir la información del usuario a expedir las sugerencias finales para la implementación. Inicialmente se consideró que el usuario seleccionara etiquetas flotantes con características de diferentes negocios, pero finalmente se terminó decantando por un acercamiento secuencial por medio de formularios, en donde al usuario se le presentaban 5 fases de manera

independiente; pero, una vez el mismo fuese seleccionado sus respuestas, la siguiente fase se ajustara a los resultados de la anterior. Para lo cual, se diseñó el siguiente boceto que obedece a las decisiones y las consecuciones tomadas en cuenta por el módulo de análisis.

Figura 5. Motor de inferencia.



Con base a las decisiones y consecuciones anteriores, se realizaron las pantallas informativas que corresponden a las sugerencias finales, y basadas en estas, las pantallas informativas que corresponden a los módulos de Pruebas y Seguridad respectivamente. La implementación de las mismas fue lo más sencillo al simplemente tener que dirigir a las pantallas prediseñadas sin mediar la intervención del usuario en alguna decisión, simplemente proveyendo el resultado deseado.

- **Fase de seguimiento y control.**

  Se realizaron las pruebas pertinentes al sistema S.A.S: Testeo de caja negra y caja blanca, corrección de errores y bugs, reasignación de tiempos a las actividades y, aplicando la metodología, se dio avance a los tres Sprint propuestos llevando a cabalidad las actividades dentro de los límites de tiempo considerados previamente.

- **Fase de cierre.**

  Cuando se consideró que los objetivos y las actividades han sido completadas con éxito, dentro de los parámetros de calidad considerados aceptables luego de las pruebas al sistema, se puso en marcha de forma exitosa el software en su versión 1.0, concluyendo con la correcta disposición de los entregables pertinentes.

3.2 TIPOS DE INVESTIGACIÓN

- **Descriptiva.**

  Se observó las necesidades actuales para con los sistemas de seguridad ofrecidos para plataformas móviles, no solo en cuestión del dispositivo físico sino también por parte de los desarrolladores, empresarios o personas naturales que, por medio de una aplicación orientada al SO Android, ofrecen una plataforma o producto que, de una u otro forma, recolecta información de los usuarios para proveer un servicio transaccional. Tomando en cuenta

los distintos tipos de negocios y las reglas asociadas a estos, las limitaciones de tiempo y dinero por parte de los emprendedores, y en algunos casos la mala asignación de recursos a una etapa del desarrollo considerada crítica como lo es la seguridad del sistema y la información.

INGENIERÍA DEL DESARROLLO DE SOFTWARE Y LA SEGURIDAD DE LA INFORMACIÓN

3.2.1 Requerimientos del nuevo sistema.

Tabla 5. Requerimientos funcionales.

CÓDIGO	DESCRIPCIÓN	REQUERIMIENTO	ACTOR
CSA1	La aplicación debe ser compatible con la versión más reciente de Android SO.	Compatibilidad.	Aplicación.
CSA2	La aplicación debe permitir al usuario elegir los labels[20] en secuencia que, en conjunción, emulen un sistema lo más similar posible al real. También el usuario debe poder deseleccionarlos.	Recolección de información.	Usuario y aplicación.
CSA3	El sistema debe realizar una captura de los labels seleccionados y, con una consulta a su base de datos, arrojar las sugerencias de seguridad pertinentes según las reglas de iteración, con la información asociada al sistema o sistemas sugeridos.	Generar las sugerencias.	Aplicación.

[20] Entiéndase los "Labels" como las etiquetas con las características de los negocios que se han considerado son más críticas. Estas generan la parte de personalización de la aplicación.

CSA4	El sistema debe permitir vaciar la sugerencia anterior y realizar un nuevo análisis sobre un sistema con características diferentes.	Reiniciar análisis.	Usuario y aplicación.
CSA5	El sistema debe almacenar la sugerencia como una imagen en el dispositivo del usuario si este lo desea.	Guardar sugerencia.	Usuario y aplicación.
CSA6	Si el usuario lo desea, podrá seleccionar unos negocios anteriormente configurados en la memoria de la aplicación a manera de “Selecciones comunes”.	Negocios comunes.	Usuario y aplicación.
CSA7	El sistema deberá presentar al usuario la opción de visualizar en forma de presentación un ejemplo de los sistemas de seguridad sugeridos luego del análisis, anteriormente implementados en un entorno seguro. Exponiendo sus características y el razonamiento que le llevó a sugerirlo según el análisis de las características provistas por el usuario.	Entorno de prueba.	Usuario y aplicación.

Requerimientos no funcionales:

- La aplicación debe ser interactiva y fácil de usar.
- La aplicación debe contar con control de excepciones.
- El sistema debe reiniciarse cada vez que se cierra la aplicación (no aplica para cuando se deja en segundo plano).
- El tiempo en que el usuario aprenda a usar el sistema debe ser menor a dos (2) horas.
- El sistema debe contar con un manual de usuario y manual técnico.

3.3 METODOLOGÍA DESARROLLO DEL SISTEMA

Para el desarrollo del proyecto se seleccionó la metodología ágil SCRUM debido a que, según lo anteriormente citado en las bases teóricas, se tendrá un alcance mayor y totalmente abierto, dando así una gran capacidad de adaptación al cambio; aunado a la interacción estrecha con el cliente (retrospectiva) para entregar un producto totalmente adaptado que cubra todas las necesidades encontradas.

3.3.1 Definición de roles

- **SCRUM Master, Miguel Peñata.** Encargado de liderar el desarrollo, el diseño y la implementación con el objetivo de cumplir a cabalidad las tareas durante los tiempos presupuestados.
- **Product Owner, Retrospectivo.** Al no tener un cliente definido, se ha optado por encarar los sprint una vez terminado con la visual de retrospectiva autocrítica con la aplicación, los resultados y las pruebas.

  El objetivo del PO, es responder a la pregunta “¿Qué hay que hacer?”, priorizando y planificando el siguiente Sprint de acuerdo a los resultados.
- **Product Backlog:** Es una documentación a manera de resultados, donde se evalúa lo que se ha hecho y se plasma lo siguiente a realizar de manera ideal.

Es el conjunto de requisitos denominados en el proceso como historias de usuario, descritos en un lenguaje sin tecnicismos. (Engineers, 2013)

Al realizar la revisión retrospectiva a modo de Product Backlog, se realizó la definición de las historias de usuario como se evidencia en las siguientes gráficas:

3.3.2 Historias de usuario

HU-01: Ingreso
Cómo: Usuario de la app.
Quiero: Ingresar al menú.
Para: Usar los módulos.

HU-02: Módulo 1
Cómo: Usuario de la app.
Quiero: Ingresar características de app.
Para: Recibir sugerencias de seguridad.

HU-03: Enseñar
Cómo: En la base de datos.
Quiero: Agregar nueva sugerencia como usuario.
Para: Agregar información más precisa.

HU-04: Módulo 2
Cómo: Usuario de la app.
Quiero: Seleccionar opciones predefinidas.
Para: Recibir sugerencias de seguridad.

HU-05: Módulo 3
Cómo: Usuario de la app.
Quiero: Leer toda la información sobre seguridad.
Para: Ampliar conocimientos

HU-06: Manual
Cómo: Usuario de la app.
Quiero: Conocer cómo funciona la app.
Para: Usarla de forma correcta.

HU-07: Guardar
Cómo: En teléfono.
Quiero: Guardar sugerencia.
Para: Recibir producto deseado.

HU-08: Info
Cómo: En base de datos.
Quiero: Guardar información del usuario.
Para: Registro.

HU-09: Pedir info
Cómo: Cuando “Enseñar” o “Guardar”. En base de datos.
Quiero: Pedir info del usuario.
Para: Registro.

- **Sprint backlog:** Es un subconjunto de ítems del Product Backlog, que son seleccionados por el equipo para realizar durante el Sprint sobre el que se va a trabajar. El equipo establece la duración de cada Sprint. Los avances de cada una de las fases en donde, presentados al docente, se sugieren los cambios pertinentes.

  Los siguientes Sprint estarán especificados en los anexos de la siguiente forma:

Tabla 6. Sprint backlog y anexos.

Sprint Backlog		
SPRINT 1	Diseño de menú	Anexo A
SPRINT 2	Desarrollo de módulo de Sugerencias	Anexo B
SPRINT 3	Bloque de enseñanza	Anexo C
SPRINT 4	Opciones predefinidas	Anexo D
SPRINT 5	Repositorio de información	Anexo E
SPRINT 6	Guardado	Anexo F

3.4 DISEÑO ARQUITECTÓNICO.

Consta de 2 capas: la capa de presentación o interfaz de usuario que interactúa directamente con el cliente y le permite el acceso a las funciones de la aplicación en forma de los 3 módulos que la componen; y la capa lógica que ejecuta las tareas asignadas a cada una de estos.

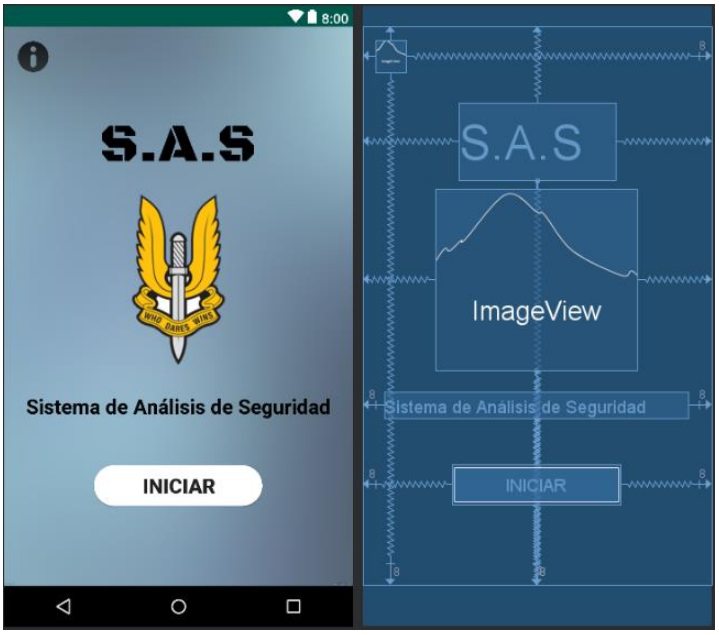
3.4.1 Funcionalidad de las capas

- **Capa de presentación:** Esta capa está formada por la interfaz de usuario en forma de menús, botones, íconos interactivos y desplegables. En esta es donde el usuario elegirá la forma de la que quiere usar la aplicación, el módulo que usará y las características que ingresará a través de su selección en los formularios secuenciales.
- **Capa lógica:** Aquí se realizan las llamadas a las diferentes pantallas que se presentarán en la interfaz de usuario, el mapeado de las distintas posibles selecciones de características, su análisis y su salida. Aquí se encuentra la parte de “Análisis de la Seguridad” por medio de los labels y la personalización en selección del usuario.

3.5 DISEÑO DE INTERFACE.

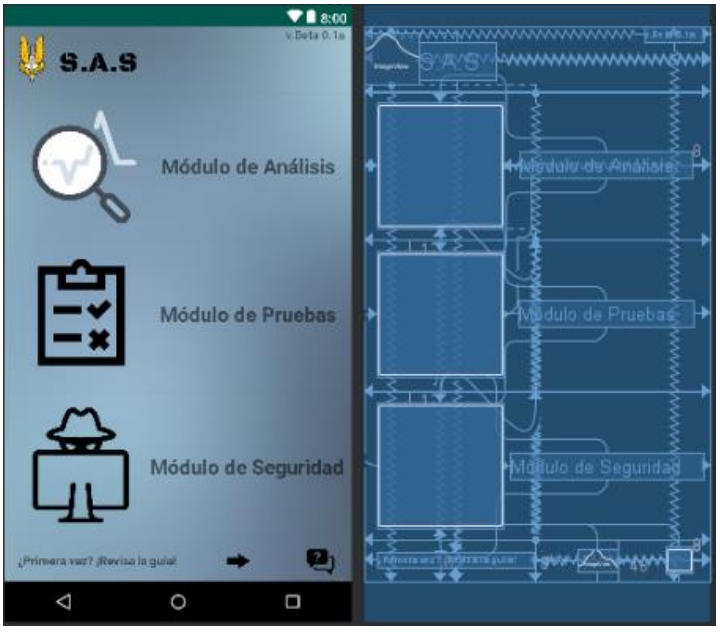
Debido a la naturaleza de código abierto, que esta totalmente disponible para su uso libre, se optó por no implementar método de control de seguridad alguno, creando en su lugar una pantalla de presentación con información sobre la versión, su autoría, fecha y el botón para inicializar. Basadas en los diseños primerizos previamente presentados como Mockups.

Figura 6. Pantalla de inicio.



Posterior a la pantalla de presentación (inicio) se despliega el menú principal con los tres módulos principales más la guía de inicio en forma de manual de usuario.

Figura 7. Menú principal.



3.6 DISEÑO DE SEGURIDAD Y CONTROLES

Desde su concepción, el sistema de análisis de la seguridad S.A.S, no se ha considerado para almacenar información delicada sobre el usuario. No pedirá información personal ni implementará métodos en segundo plano que requieran una escala en la seguridad y solicitud de permisos. La aplicación no hace absolutamente nada sin la previa orden del usuario.

La seguridad está asegurada desde la naturaleza inofensiva de la aplicación, la cual busca simplemente compartir el conocimiento almacenado luego de la investigación a las debilidades y fortalezas de los sistemas, y una alteración de esta información no es posible de primera mano debido a que la misma aplicación no la almacena de forma virtual asequible, sino en forma de renders del sistema.

3.7 SELECCIÓN DE HERRAMIENTAS DE DESARROLLO Y PROGRAMACIÓN

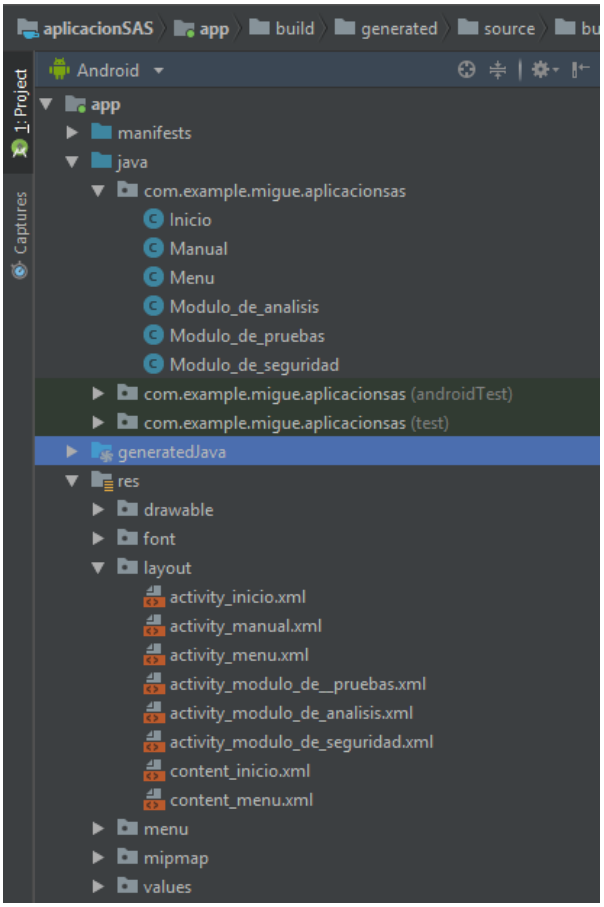
Para la realización del presente proyecto se hizo uso de las herramientas de desarrollo Android Studio, con un lenguaje base de programación en JAVA para la capa lógica de la aplicación, y XML para la interfaz gráfica. Se usó la versión 28 del API de ANDROID para el despliegue y testeo de la plataforma virtual, y un dispositivo SAMSUNG SM-J700M con versión de API 23 de ANDROID 6.0.1 para el despliegue real y las pruebas en entorno real.

4. ANÁLISIS DE RESULTADOS Y CONCLUSIONES

4.1 CODIFICACIÓN DEL PROGRAMA

En esta instancia se especifican los componentes de frontend y backend (listados de forma no exhaustiva) de la aplicación. Mostrados según su desarrollo en Android Studio, omitiendo los apartados funcionales del S.O y centrándose solamente en la funcionalidad general del Sistema S.A.S. Para mayor información dirigirse a los anexos G y H, los manuales de usuario y técnico respectivamente.

Figura 8. Componentes programáticos del aplicativo[21].



[21] Se han omitido gran cantidad de instancias y documentos tanto .xml como .java, con la intención de evitar redundantes y especificar directamente las instancias esenciales del núcleo.

Tabla 7. Componentes programáticos del aplicativo (Descriptivo).

Programa	Descripción	Proceso Afectado	Disparador
Inicio	Programa que maneja y muestra el *layout* de la página de bienvenida, con el logo del programa y el botón para inicializar el menú.	Inicio	Automático
Manual	Subinstancia del módulo de seguridad, también referido como el módulo de información a pesar de no ser un módulo como tal. Contiene información sobre el programa: la licencia, el autor, el título y la fecha.	Inicio – Módulos	Inicio – Automático
Menu	Pantalla que llama el *layout* que contiene los botones para iniciar cada uno de los módulos.	Módulos	Automático – Usuario
Modulo_de_analisis	Módulo principal que contiene el motor de inferencia, el formulario de customización y llama los *layout* de información y sugerencias; luego de aplicar los limitantes y	Módulos – Submódulo de guardado	Usuario

	según la información provista por el usuario.		
Modulo_de_pruebas	Módulo gemelo y que funciona en paralelo al módulo de análisis. Funciona, de igual manera, llamando los *layout* de información y sugerencias, pero lo hace bajo preselecciones y no bajo personalización del usuario.	Módulos – Submódulo de guardado	Usuario
Modulo_de_seguridad	Contiene las llamadas a todos los *layout* que contienen la información completa que se ha volcado sobre el sistema. De libre acceso por el usuario en cualquier momento.	Módulos	Usuario
layouts	Pantallas estáticas y/o interactivas que conforman el grueso del front-end del programa. Contienen la información de la seguridad y las sugerencias, así como el formulario y el submódulo de guardado.	Inicio – Módulos – Submódulo de guardado	Automático

4.2 PRUEBAS.

4.2.1 Pruebas de función

Con esta prueba se verifica la correcta navegación por los módulos y los *layout*, así como también se verifica la correcta visualización de los mismos, y se garantiza el funcionamiento de los botones, el formulario, el submódulo de guardado y los módulos.

Tabla 8. Prueba de caja blanca.

Tipo: Caja Blanca		
Módulos probados: **Todos**		
ENTRADA	**PROCESO**	**SALIDA**
Navegación e interacción con los *layout*.	Se verifica la integridad estructural de la aplicación, la calidad de las visualizaciones y la facilidad de acceso.	Se cargan los módulos y se cierran, se llaman los *layout*.

Una vez concluidos las pruebas de caja blanca, se prosiguió a contactar un *tester* externo sin ninguna relación con el desarrollo de la aplicación para llevar a cabo las pruebas de funcionalidad de caja negra y la interfaz.

Tabla 9. Prueba de caja negra – Beta testing.

Tipo: Caja Negra			
TIPO	**MÓDULO**	**PROCEDIMIENTO**	**RESULTADO**
Beta testing.	Todos.	Navegación, interacción, retroceso y avance a través de los botones de la aplicación y la funcionalidad *back* de Android. Interacción completa con los módulos y submódulos. Comprensión de lectura y análisis de la información de seguridad suministrada por la aplicación.	ACEPTABLE. PEQUEÑAS CORRECCIONES REALIZADAS.
OBSERVACIONES			
El estado funcional de la aplicación fue hallado en óptimas condiciones. Las sugerencias dadas por el *tester*, y posteriormente tomadas en cuenta e implementadas, referían redacción y complejidad de algunos conceptos al momento de presentarlos al usuario.			

4.2.2 Pruebas modulares

Estas pruebas permitieron verificar la integridad y la operabilidad de los diferentes módulos que conforman el aplicativo. Debido a la naturaleza integral del mismo, se consideró que las mismas fueron realizadas de manera exhaustiva durante las pruebas de funcionalidad.

4.2.3 Pruebas del sistema

Este tipo de pruebas se efectuó para evaluar el desempeño general de la aplicación y el sistema. Comprende los siguientes pasos:

- Prueba de integración: Cada módulo está en relación con otro(s), se probaron independientemente y luego se realizo una prueba integral del sistema.
- Pruebas de rendimiento: Se verificó la ejecución de cada una de las instancias del programa, así como el despliegue y la correcta redirección por parte de los vínculos, además se realizaron pruebas de rendimiento.
- Pruebas de consistencia: Se realizaron pruebas de consistencia en cada uno de los módulos, durante la ejecución del programa, además se actualizaron cada uno de los módulos del aplicativo según los resultados se iban dando, hasta obtener los resultados deseados.

4.2.4 Pruebas de interfaz

Debido a la naturaleza del aplicativo, las pruebas de interfaz requerían un exhaustivo proceso de verificación individual de cada uno de los *layout* que comprendían el frontend del programa, así como verificar la integridad de la información contenida y mostrada en ellos. Esta prueba se consideró realizada luego de las pruebas de función y modulares.

4.2.5 Pruebas de calidad

Tomando como punto de enfoque las normas sobre la calidad y gestión de calidad, establecidas por la Organización Internacional de Normatilización ISO, se enfocó el proyecto desde su concepción con base a los siguientes principios:

1. **Enfoque al cliente:** Desde el día 1, se clarificó que uno de los objetivos principales del proyecto era fomentar el conocimiento sobre la seguridad de forma fácil, sencilla y al alcance de quien quiera obtenerlo.
2. **Liderazgo:** Utilizando una metodología de desarrollo ágil y planificando todos y cada uno de los pasos a seguir, se realizó un proceso liderado con seriedad, profesionalidad y pasión; a pesar de contar con una sola persona. Aún así, se consideró que el liderazgo era de vital importancia para el desarrollo sobresaliente de la aplicación.
3. **Participación del personal:** Tomando al personal tanto como a quienes colaboraron con el proyecto de manera directa o indirecta, tanto personas como herramientas, se considera que los recursos invertidos en el desarrollo del proyecto satisfacen la participación completa y el total uso de los mismos.
4. **Enfoque basado en procesos:** El proceso detallado durante el cronograma de actividades y la metodología de desarrollo ágil SCRUM, denota un enfoque totalmente centrado en un proceso de desarrollo profesional.

La norma cuenta con más principios, pero los mismos ya no se consideran aplicables para el entorno en el cual se ha desarrollado la aplicación S.A.S.

4.2.6 Informe general

Tabla 10. Informe general de pruebas.

Tipo de pruebas generales	SÍ Cumple	No Cumple
Acceso al sistema de acuerdo con los parámetros definidos.	**X**	
Acceso a cada uno de los módulos que conforman el sistema, de forma directa o indirecta.	**X**	
Navegación hacia adelante y hacia atrás dentro del sistema.	**X**	
Acceso a los manuales de ayuda.	**X**	
Pruebas de integración.	**X**	
Pruebas modulares.	**X**	
Pruebas de rendimiento.	**X**	
Pruebas de calidad.	**X**	
Pruebas del sistema.	**X**	

4.3 CONCLUSIONES

- Trabajando con los tiempos que se estipularon y para los cuales se elaboró un plan de desarrollo utilizando la metodología ágil SCRUM, se diseño e implementó una aplicación capaz de recibir las características de un sistema a través de su módulo principal y arrojar las sugerencias de seguridad que mejor se ajustan a las características del mismo.
- Se realizó una investigación a través de distintas bases de datos y estudios sobre la seguridad de la cual se extrajeron y se implementaron en las sugerencias finales, los sistemas de seguridad que mejor ajustan las falencias de acuerdo a las características de los distintos sistemas. Y se implementaron en el módulo de análisis, los valores preestablecidos que funcionan como “elecciones populares”.
- A través de un formulario secuencial que se dividió en 4 fases y unos limitantes, se implementó un sistema de entrada que podía captar la información que el usuario provee sobre su sistema.
- Utilizando Photoshop y XML, se diseñó de manera satisfactoria una interfaz de usuario que contiene los accesos, menús, íconos y la información que posteriormente se montó sobre la aplicación.
- Al finalizar el proyecto, se diseñaron e implementaron la versión vBeta.0.1a que consistía solamente de la capa de presentación. Y posteriormente se concluyó con la entrega satisfactoria de la versión vTest.1.0 que contenía las capas de presentación y lógica, con todos los módulos que se plantearon, funcionales.

4.4 RECOMENDACIONES

- El código del programa es totalmente libre, el manual técnico incluye el diagrama de toma de decisiones que se puede usar a manera de guía sobre el formulario de secuencia. Se recomienda agregar y modificar la información sobre los sistemas de seguridad según estos se vayan actualizando y modificar la secuencia de la toma de decisiones según sea conveniente manteniendo la integridad de la información y primando la enseñanza sobre los sistemas de seguridad por sobre la seguridad de la aplicación misma.

5. BIBLIOGRAFÍA

- ©2018 Google, español. Resultados de búsqueda en aplicaciones en todos los idiomas, precios y valoraciones. Obtenido de *https://play.google.com/store/search?q=security&c=apps&hl=es*
- L. Andrew and A. Carmela. 2013 mobile future in focus, January 2013.
- Según J. Park, B. Kim, S.-R. Kim, J. H. Yoo., y Y. Cho. En su Performance analysis of security enforcement on Android operating system. En Proceedings of the 2011 ACM Symposium on Research in Applied Computation, RACS '11, páginas 282-286, New York, NY, USA, 2011. ACM.
- S. Brahler. Analysis of the Android Architecture. PhD thesis, Karlsruher Institut fur Technologie, 2010.
- *https://letsknowaboutandroid.wordpress.com/about/* (s.f.) Obtenido de https://letsknowaboutandroid.wordpress.com/about/
- A. Alfonso. Informe de seguridad en iOS y Android, abril 2012.
- H. Bing. Analysis and research of system security based on Android. En Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference en, páginas 581-584, 2012.
- A. Alfonso. Informe de seguridad en iOS y Android, abril 2012.
- A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, y C. Glezer. Google Android: A comprehensive security assessment. Security Privacy, IEEE, 8(2):35-44, 2010.
- *http://source.android.com/tech/security/* (s.f.) Obtenido de http://source.android.com/tech/security/
- A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, y C. Glezer. Google Android: A comprehensive security assessment. Security Privacy, IEEE, 8(2):35-44, 2010.
- Ministerio del Interior y el Fondo para la Participación y el Fortalecimiento de la Democracia. (2014). Políticas para la Gestión de Seguridad de la Información Control de Acceso. Obtenido de:

https://www.mininterior.gov.co/sites/default/files/oip-2014-psi-especificas-6_control_acceso.doc

- Information Security in Banking and Financial Industry. IJCEM International Journal of Computational Engineering & Management, vol. 14, October 2011. Obtenido de: *http://www.IJCEM.org. Fuente principal.*
- Online Banking Report. The Online Banking Report, 1999. Obtenido de *http://www.onlinebakingreport.com.*
- Principles of transaction-oriented database recovery. ACM Computing Surveys. Haerder, Reuter. 15(4): Punto 287.
- Mobile Banking Operation System. Obtenido de: *semanticscholar.org/3475/8c819d3bb4d4e5e4b4d053db523684adac09.pdf*
- Interfaces (presentation). The user interface of mechanical system, a vehicle or an industrial installation is sometimes referred to as the human-machine interface (HMI). Ben Griffin, Laurel Baston. 2014.
- Utilización de UML en Ingeniería del Software con Objetos y Componentes. Stevens, Pooley, Wesley. 2002.
- Para qué sirve el SCRUM en la metodología ágil. (s.f.) Obtenido de: *http://www.i2btech.com/blog-i2b/tech-deployment/para-que-sirve-el-scrum-en-la-metodologia-agil/*
- ISO 9000 Organización Internacional para la Estandarización (ISO). Gestión de calidad. (s.f) Obtenido de: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=46486
- A Conceptual Exploration for the Safe Development of Mobile Devices Software Based on OWASP. (s.f.) Fundación Universitaria los Libertadores, Bogotá D.C., Colombia. Gil, Baquero, Plazas, Hernández. 2018.
- OWASP. (18 de 07 de 2016). Proyecto OWASP Mobile Security. Obtenido de https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

- OWASP. (2016). Modelado de Amenazas. Obtenido de https://www.owasp.org/index.php/Modelado_de_Amenazas.
- OWASP. (2016). Threat Risk Modeling. Obtenido de https://www.owasp.org/index.php/Threat_Risk_Modeling#DREAD
- SEGURINFO. (25 de 10 de 2016). Los 10 errores más comunes de seguridad de aplicaciones móviles. Obtenido de http://www.segurinfo.org/detalle.php?a=los-10-errores-mas-comunes-de-seguridad-de-aplicaciones-moviles&t=2&d=542

ANEXOS

Sprint del punto 3.3 Diseño Metodológico. Sprint backlog organizados según el orden de desarrollo y el historial de usuario.

- **ANEXO A**

SPRINT 1: Diseño de menú.

Tareas pendientes	0
Horas estimadas	25
Horas ejecutadas	23

Historial de usuario	**Tarea**	**Tipo**	**Estado**	**Estimación (horas)**	**Esfuerzo**					**Esfuerzo total**
					Día 1	**Día 2**	**Día 3**	**Día 4**	**Día 5**	
HU-01 Quiero ingresar al menú para usar los módulos	Análisis de requerimientos	Análisis	Terminada	5	5					5
	Diseño	Diseño	Terminada	10		5	3			8
	Prototipado	Desarrollo	Terminada	10				5	5	10

- **ANEXO B**

SPRINT 2: Desarrollo de módulo de sugerencias.

Tareas pendientes	0
Horas estimadas	50
Horas ejecutadas	48

Historial de usuario	Tarea	Tipo	Estado	Horas est.	Esfuerzo										Esfuerzo total
					Día 1	Día 2	Día 3	Día 4	Día 5	Día 6	Día 7	Día 8	Día 9	Día 10	
HU-02 Quiero ingresar características de mi app para recibir sugerencias de seguridad	Diseño de formulario	Diseño	Terminada	10	5	5									10
	Diseño de base de datos	Diseño	Terminada	10			5	5							10
	Conexión de base de datos con formulario	Desarrollo	Terminada	15					5	5	5				15
	Implementación de sugerencias	Desarrollo	Terminada	10								5	3		8
	Pruebas	Testing	Terminada	5									2	3	5

- **ANEXO C**

SPRINT 3: Bloque de enseñanza.

Tareas pendientes	0
Horas estimadas	30
Horas ejecutadas	30

Historial de usuario	**Tarea**	**Tipo**	**Estado**	**Estimación (horas)**	**Esfuerzo**						**Esfuerzo total**
					Día 1	**Día 2**	**Día 3**	**Día 4**	**Día 5**	**Día 6**	
HU-03 Quiero agregar nueva sugerencia como usuario para agregar información más precisa	Reformulación del formulario	Diseño	Terminada	5	5						5
	Desarrollo del nuevo formulario para agregar información	Desarrollo	Terminada	15		5	5	5			15
	Conexión a base de datos	Desarrollo	Terminada	5					5		5
	Testeo	Testing	Terminada	5						5	5

- **ANEXO D**

SPRINT 4: Opciones predefinidas.

Tareas pendientes	0
Horas estimadas	20
Horas ejecutadas	20

Historial de usuario	Tarea	Tipo	Estado	Estimación (horas)	Esfuerzo					Esfuerzo total
					Día 1	Día 2	Día 3	Día 4	Día 5	
HU-04 Quiero seleccionar opciones predefinidas para recibir sugerencias de seguridad	Implementación de opciones	Desarrollo	Terminada	10	5	5				10
	Conexión a base de datos	Desarrollo	Terminada	5			5			5
	Pruebas	Testing	Terminada	5				5	0	5

- **ANEXO E**

SPRINT 5: Repositorio de información.

Tareas pendientes	0
Horas estimadas	25
Horas ejecutadas	18

Historial de usuario	Tarea	Tipo	Estado	Horas est.	Esfuerzo					Esfuerzo total
					Día 1	Día 2	Día 3	Día 4	Día 5	
HU-05 Quiero leer toda la información sobre seguridad para ampliar conocimientos	Diseño de opciones	Diseño	Terminada	10	5	3				8
	Conexión con base de datos	Diseño	Terminada	10		2	5			5
HU-06 Quiero conocer cómo funciona la app para usarla de forma correcta	Desarrollo del manual	Desarrollo	Terminada	5				5	0	5

- **ANEXO F**

SPRINT 6: Guardado.

Tareas pendientes	0
Horas estimadas	40
Horas ejecutadas	40

Historial de usuario	Tarea	Tipo	Estado	Horas est.	Esfuerzo										Esfuerzo total
					Día 1	Día 2	Día 3	Día 4	Día 5	Día 5	Día 5	Día 5	Día 5	Día 5	
HU-07 Quiero guardar sugerencia para recibir producto deseado	Captura de información de base de datos	Desarrollo	Terminada	10	5	5									10
	Modo de guardado en teléfono	Desarrollo	Terminada	10			5	5							10
HU-08 Quiero guardar información del usuario para registro	Modo de guardado en base de datos de información del usuario	Desarrollo	Terminada	10					5	5					10
HU-09 Quiero pedir información del usuario para registro	Desarrollo de formulario para información del usuario	Desarrollo	Terminada	10							5	5	0	0	10

- **ANEXO G**

MANUAL DE USUARIO

INTRODUCCIÓN

El sistema de análisis de seguridad PRS Móvil está diseñado para aplicaciones desarrolladas para plataformas móviles Android. El mismo cuenta con distintos módulos que le facilitan al usuario obtener sus resultados: la obtención de una recomendación para la implementación de un sistema de seguridad para una aplicación móvil desarrollada para el sistema operativo Android.

Es de acceso libre, pensado para usuarios con conocimientos de programación móvil o web que deseen expandir sus conocimientos para con los sistemas de seguridad, normas, medidas, recomendaciones; así también como las medidas a llevar a cabo para combatir las vulnerabilidades y amenazas que enfrentan los sistemas de seguridad de hoy en día.

Las recomendaciones provienen de diferentes fuentes, principalmente documentación basada en las investigaciones de OWASP, en cuánto a aplicaciones web y móviles.

OBJETIVOS DEL SISTEMA

- Arrojas sugerencias personalizadas para la implementación de sistemas de seguridad que se ajusten a las necesidades del usuario.

- Facilitar a los nuevos empresarios e impulsadores la accesibilidad a sistemas de seguridad acordes a sus necesidades, y, por ende, fomentar la implementación de sistemas más seguros.

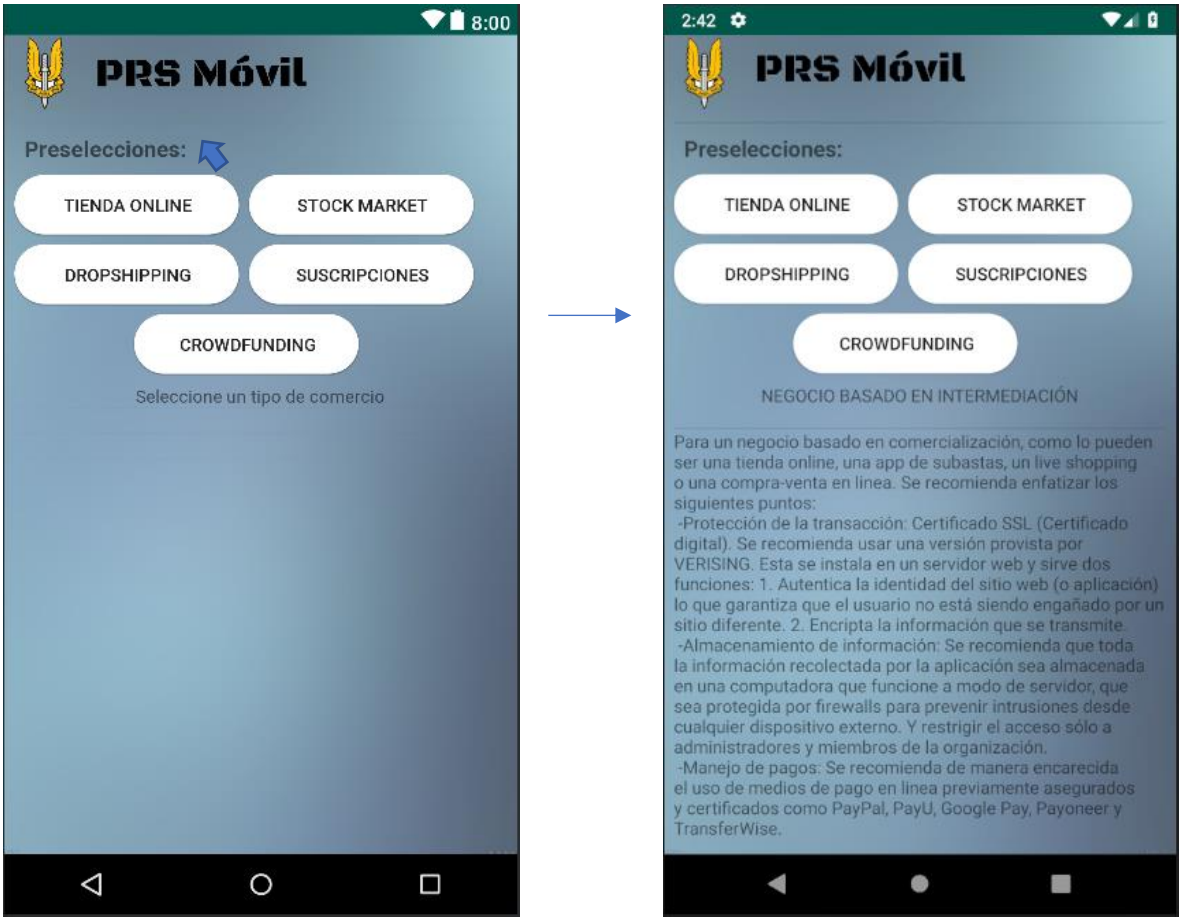
- Presentar al público el conocimiento básico sobre sistemas de seguridad, sus usos más comunes y las vulnerabilidades que afrontan.

- Fomentar la buena praxis para con el desarrollo preliminar de plataformas móviles en cuanto a su seguridad, reduciendo el número de estafas y brindando un poco de luz sobre un tema al cual se le presta poca atención y sobre el cual se debería invertir más; pero, también invertir bien.

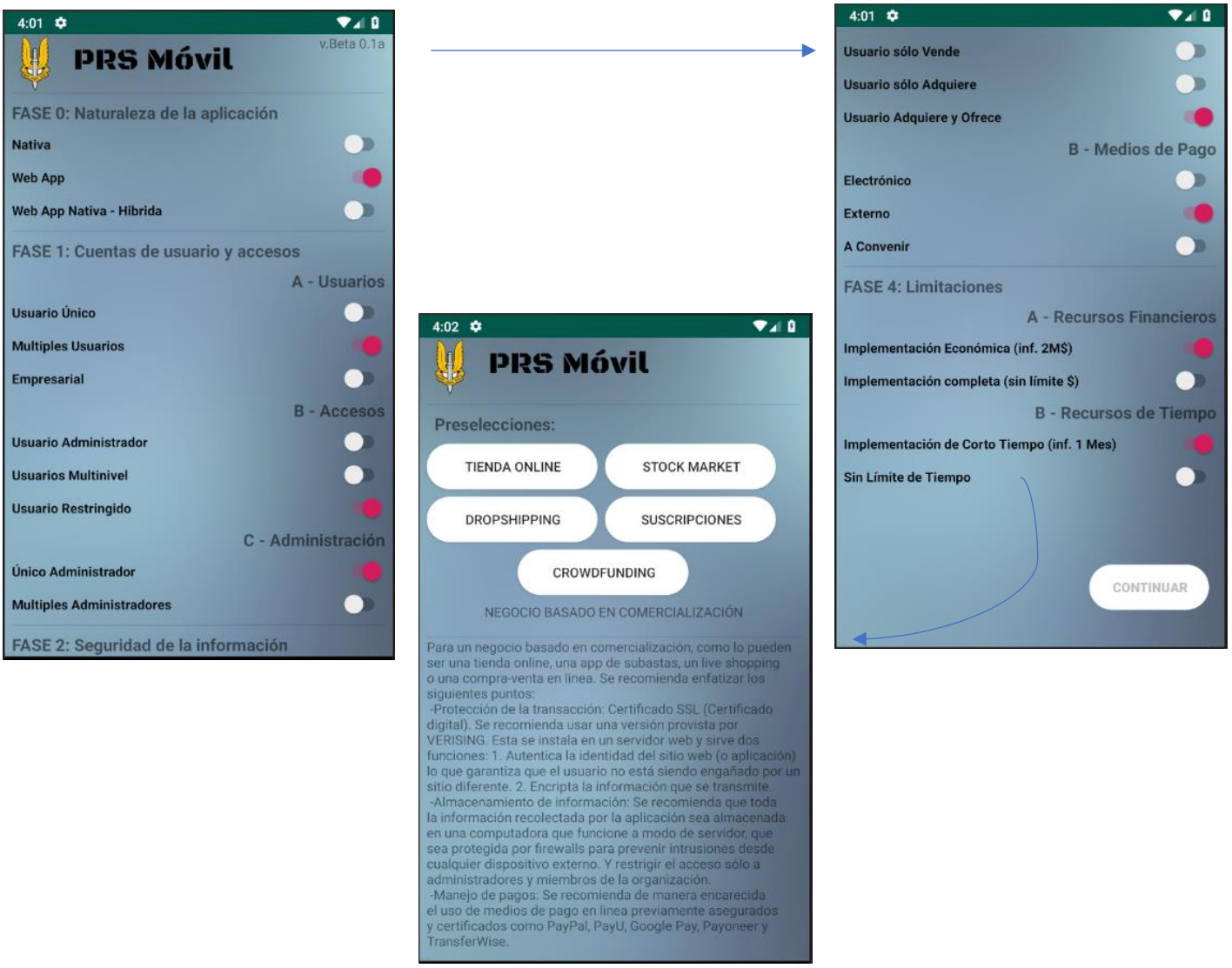
GUÍA DE USO

1- Módulo de Preselecciones.

El módulo de preselección es bastante directo, simplemente seleccionar el modelo de negocio y se obtiene la recomendación de seguridad acorde al botón seleccionado:



Para utilizar el módulo de análisis basta rellenar el formulario de respuesta única por inciso. Todos los campos han de estar rellenados para que el programa pueda acotar el tipo de comercio que mejor se ajusta al tipo de aplicación analizada. Una vez se haya rellenado, el botón de análisis se activa y redirige al módulo de pruebas donde mostrará la recomendación.



3- Módulo de Seguridad.

El módulo de seguridad no tiene uso activo alguno. Su ingreso es suficiente para obtener la información que el mismo enseña. Tomada de Le VPN en cuanto a las vulnerabilidades más comunes de las aplicaciones móviles.

SOPORTE

Para mayor información comunicarse a la línea 47955225. O al correo: mapenatar@libertadores.edu.co.

- **ANEXO H**

MANUAL TÉCNICO

OBJETIVOS Y ALCANCE

OBJETIVOS

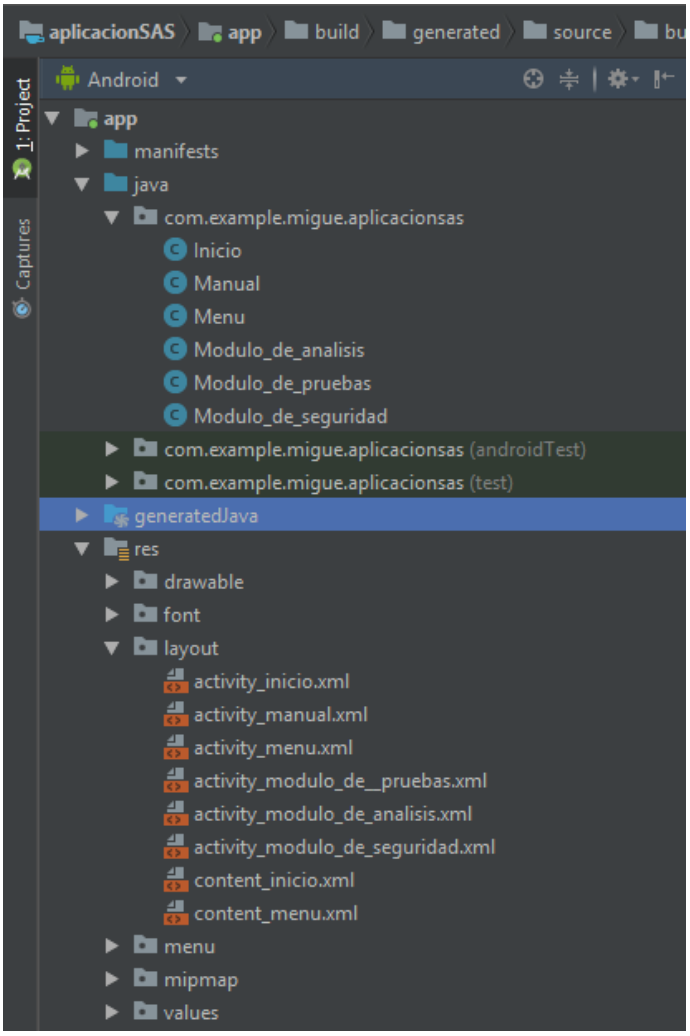
- Iterar las vulnerabilidades con los sistemas de seguridad que se enfoquen a dichas falencias.
- Implementar un formulario secuencial que tome las distintas posibilidades ingresadas por el usuario en forma de características, arrojando los resultados más adecuados para dichos escenarios.
- Listar una serie de valores preestablecidos que funcionarán como “Mejores elecciones/Elecciones populares” con base en los resultados de las pruebas con las vulnerabilidades y necesidades más comunes.

ALCANCE

La app ha sido desarrollada como una aplicación móvil para tecnologías Android, minimizando a cero la intrusión a la información almacenada en dispositivo host con la intención de conservar la privacidad del usuario.

DESCRIPCIÓN DEL CÓDIGO

Figura 1. Estructura programática.



Los componentes programáticos del aplicativo en Android están divididos en dos grandes partes que se comunican y complementan. Los modelos xml de layouts que compone el front-end de la aplicación y la codificación en Java que compone el back-end. Ambas partes componen la capa de negocios de la aplicación, que se comunica por medio de una conexión sencilla al servidor de la base de datos, provisto por Google Firebase, en su modelo de actualización en tiempo real.

Figura 2. Base de datos

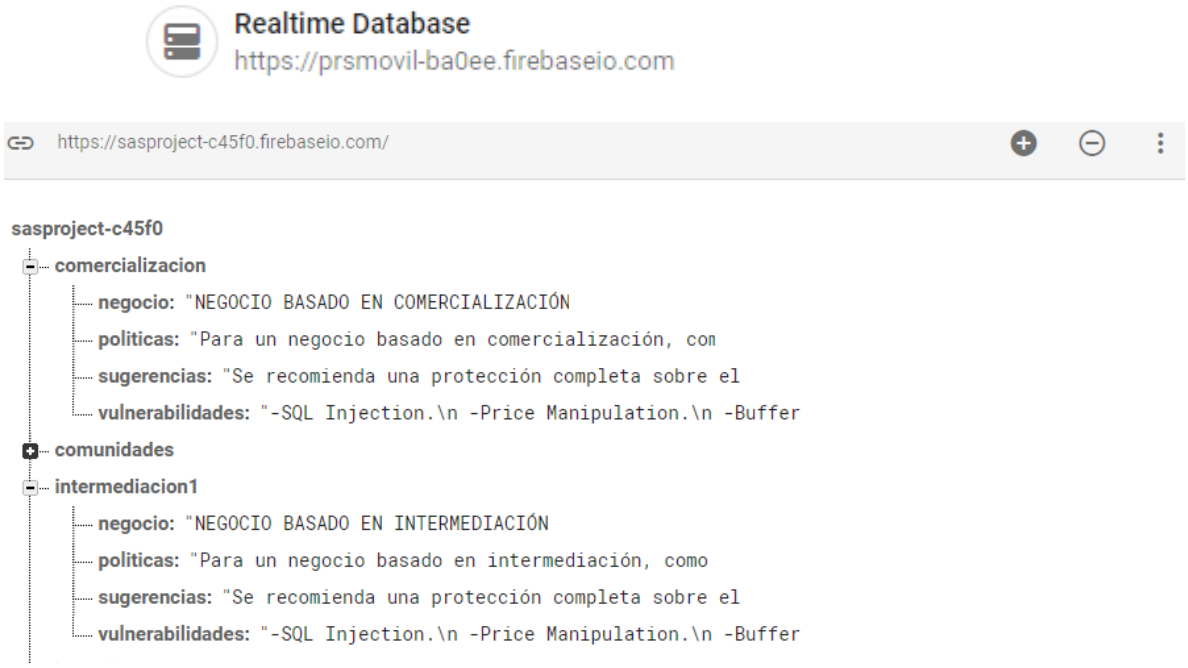


Tabla 1. Componentes programáticos del aplicativo (Descriptivo).

Programa	Descripción	Proceso Afectado	Disparador
Inicio	Programa que maneja y muestra el *layout* de la página de bienvenida, con el logo del programa y el botón para inicializar el menú.	Inicio	Automático
Manual	Subinstancia del módulo de seguridad, también referido como el módulo de información a pesar de no ser un módulo como tal. Contiene información sobre el programa: la licencia, el autor, el título y la fecha.	Inicio – Módulos	Inicio – Automático

Menu	Pantalla que llama el *layout* que contiene los botones para iniciar cada uno de los módulos.	Módulos	Automático – Usuario
Modulo_de_analisis	Módulo principal que contiene el motor de inferencia, el formulario de customización y llama los *layout* de información y sugerencias; luego de aplicar los limitantes y según la información provista por el usuario.	Módulos – Submódulo de guardado	Usuario
Modulo_de_pruebas	Módulo gemelo y que funciona en paralelo al módulo de análisis. Funciona, de igual manera, llamando los *layout* de información y sugerencias, pero lo hace bajo preselecciones y no bajo personalización del usuario.	Módulos – Submódulo de guardado	Usuario
Modulo_de_seguridad	Contiene las llamadas a todos los *layout* que contienen la información completa que se ha volcado sobre el sistema. De libre acceso por el usuario en cualquier momento.	Módulos	Usuario
layouts	Pantallas estáticas y/o interactivas que conforman el grueso del front-end del programa. Contienen la información de la seguridad y las sugerencias, así como el formulario y el submódulo de guardado.	Inicio – Módulos – Submódulo de guardado	Automático

SOPORTE

Para mayor información comunicarse a la línea 47955225. O al correo: mapenatar@libertadores.edu.co.