Mario Alexander Velasco **Ingeniero de Sistemas**

Universidad de
PAMPLONA
La Academia al servicio de la Vida

UNIVERSIDAD DE PAMPLONA
FACULTA DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS

TRABAJO DE GRADO PRESENTADO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

TEMA:
PROPUESTA DE UN PROCEDIMIENTO PARA EL DESARROLLO DE APLICACIONES EN ANDROID

AUTOR:
MARIO ALEXANDER VELASCO

PAMPLONA, NORTE DE SANTANDER
DICIEMBRE 2016

UNIVERSIDAD DE PAMPLONA
FACULTA DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS

TRABAJO DE GRADO PRESENTADO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

TEMA:
PROPUESTA DE UN PROCEDIMIENTO PARA EL DESARROLLO DE APLICACIONES EN ANDROID

Autor:
MARIO ALEXANDER VELASCO

DIRECTOR: M.Sc. EDGAR ALEXIS ALBORNOZ ESPINEL

DIRECTOR DEL PROGRAMA: PhD. CARLOS ARTURO PARRA ORTEGA

PAMPLONA, NORTE DE SANTANDER
DICIEMBRE 2016

UNIVERSIDAD DE PAMPLONA
FACULTA DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS

TRABAJO DE GRADO PRESENTADO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

TEMA:
PROPUESTA DE UN PROCEDIMIENTO PARA EL DESARROLLO DE APLICACIONES EN ANDROID

FECHA DE INICIO DEL TRABAJO: FEBRERO DE 2016
FECHA DE TERMINACION DEL TRABAJO: DICIEMBRE DE 2016

NOMBRES Y FIRMAS DE AUTORIZACION PARA SUSTENTAR:

AUTOR: Mario Alexander Velasco Vera _____

DIRECTOR: M.Sc. Edgar Alexis Albornoz Espinel _____

DIRECTOR DE PROGRAMA: PhD. Carlos Arturo Parra Ortega _____

JURADO CALIFICADOR:

PRESIDENTE: _____

OPONENTE: _____

SECRETARIO: _____

PAMPLONA, NORTE DE SANTANDER
DICIEMBRE 2016

Mario Alexander Velasco **Ingeniero de Sistemas**

CITAS

“La separación de talento y habilidad es uno de los conceptos peor malentendidos para la gente que intenta destacar, que tiene sueños, que quiere hacer cosas. El talento lo tienes de forma natural, la habilidad solo se desarrolla por horas, horas y horas de entrenamiento.”

Will Smith

"Hay dos maneras de diseñar software: una es hacerlo tan simple que sea obvia su falta de deficiencias, y la otra es hacerlo tan complejo que no haya deficiencias obvias"

C.A.R. Hoare

“Un ordenador es para mí la herramienta más sorprendente que hayamos ideado. Es el equivalente a una bicicleta para nuestras mentes”

Steve Jobs

“El mayor enemigo del conocimiento no es la ignorancia, sino la ilusión del conocimiento”

Stephen Hawking

AGRADECIMIENTOS

A Dios por ser mi compañero en todo momento, a la Universidad de Pamplona por haberme brindado la oportunidad de cumplir una de mis metas y formarme como profesional.

A todos y cada uno de mis docentes, los cuales me brindaron su sabiduría y vivencias dentro del proceso educativo, permitiéndome avivar mis ansias por nuevos conocimientos y el alcanzar este gran logro.

A mis padres, quienes a pesar de los inconvenientes económicos me impulsaron a tomar este rumbo y se preocuparon constantemente por su culminación; por todo el tiempo que invirtieron en mí, por tantas noches de desvelo y por cada uno de sus sabios consejos para ser una persona correcta, siendo ellos mí ejemplo a seguir.

A mi hermosa mujer Sally Jazmín, quien desde el comienzo me brindo todo su amor y un apoyo incondicional en todos los aspectos de mi vida, por permanecer a mi lado aun en los momentos de oscuridad y por regalarme a lo más bello que tengo en la vida, mi hermosa hija Jessica Alejandra a quien dedico este trabajo.

A mis suegros Tairo y Rosita, los cuales me apoyaron desinteresadamente, aun sin ser su obligación, y permitieron que diera los últimos pasos a lo largo de este proceso, motivándome siempre en pro de su culminación.

A mis mejores amigos Miguel y Doyler, por compartir conmigo este proceso, y quienes junto a familiares y compañeros siempre estuvieron a mi lado con su apoyo y consejos.

Mario Alexander Velasco Ingeniero de Sistemas

RESUMEN

La presente investigación tuvo como objetivo principal establecer un procedimiento para el desarrollo de aplicaciones para el sistema operativo Android. Para lograr éste propósito se procedió a consultar y analizar las fuentes actualmente relacionadas con desarrollo software en dispositivos móviles, esta base teórica permitió identificar los métodos agiles como los más relevantes para abordar éste proyecto debido a características como su disposición al cambio, periodos cortos de tiempo, equipo de desarrollo pequeño, comunicación continua y asesoramiento del mismo cliente. Se seleccionaron las metodologías Scrum, eXtreming Programing (XP), Desarrollo orientado a pruebas (TDD) y Mobile-D como principales procedimientos aplicados a trabajos sobre implementación de aplicaciones móviles y se evaluaron sus etapas según criterios como sencillez en la aplicabilidad y gestión del proceso; y efectividad para conseguir una aplicación funcional a corto plazo y con bajos recursos.

Las características recopiladas y aquellas que fueron adaptadas permitieron la definición del procedimiento EfiSenDroid, específico para el desarrollo de aplicaciones para Android, el cual fue validado tras ser aplicado junto con el framework App Inventor 2, como herramienta de programación, en la implementación del prototipo Assistance Control de una aplicación nativa que permite llevar el listado de asistencia de determinados grupos de estudiantes, así como modificarla o exportarla como documento de texto haciendo uso del almacenamiento interno del dispositivo y sin necesidad de conexiones de red. Finalmente el análisis de los resultados aporta un alto nivel de satisfacción con respecto a la aplicabilidad del procedimiento y la obtención de un producto funcional desarrollado en un corto periodo de tiempo.

Mario Alexander Velasco Ingeniero de Sistemas

ABSTRACT

The main objective of this research was to establish a procedure for the development of applications for the Android Operating System. In order to achieve this purpose, sources currently related to software development in mobile devices were researched and analyzed; this theoretical framework allowed the identification of agile software development methodologies as the most relevant to address this project due to characteristics such as their versatility to change, short periods of time, small development team, continuous communication and advice from the customer itself. Scrum, eXtreming Programing (XP), Test Driven Development (TDD) and Mobile-D methodologies were selected as the main procedures applied to work on mobile application implementation and their stages were assessed according to criteria such as simplicity in applicability and process management; and effectiveness to achieve a short-term functional application with low resources.

The gathered features and those that were adapted allowed the definition of the EfiSenDroid procedure, specific for the development of applications for Android, which was validated after being applied together with the “App Inventor 2 framework” as a programming tool in the implementation of the prototype “Assistance Control” of a native application that allows to monitor the attendance list of specific groups of students as well as modify or export it as a text document making use of internal storage of the device and without the need for network connections. Finally, the analysis of the results provided a high level of satisfaction regarding the applicability of the procedure and obtaining a functional product developed in a short period of time.

Mario Alexander Velasco Ingeniero de Sistemas

INDICE DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

Mario Alexander Velasco **Ingeniero de Sistemas**

INTRODUCCIÓN

Actualmente Android es el sistema operativo más utilizado en dispositivos móviles (Smartphone, tablets, smartwatch, Smart TV, etc.), tanto así que aproximadamente 8 de cada 10 personas poseen algún equipo basado en éste sistema. Además también cuenta con la mayor cantidad de aplicaciones, más de 2.400.000 publicadas oficialmente en su tienda online Google Play, sin contar aquellas no oficiales desarrolladas por terceros. Android permite la creación de diferentes tipos de aplicaciones y además proporciona software oficial para la implementación de éstas, con lo cual cualquier persona podría dedicarse a programar en esta plataforma y, de hacerlo bien, llegaría a ser rentable. Sin embargo, aunque existen las herramientas el inconveniente está en el proceso, ya que no existe una estructura o método estándar establecido para la elaboración de aplicaciones móviles para el sistema operativo Android.

La presente investigación hace referencia a la definición de un procedimiento específico para desarrollar aplicaciones en Android, cuyo principal interés radica en que a partir de una idea sea posible estructurar de manera sencilla las fases y tareas necesarias para la implementación de un producto funcional en un periodo corto de tiempo, además se enfoca a proyectos con un mínimo de integrantes y recursos de modo que sea posible abordarlos de manera personal.

Para poder proponer el procedimiento fue necesario identificar y analizar diferentes tipos de fuentes sobre el desarrollo de software móvil en la actualidad, los procedimientos, metodologías y tareas utilizadas, así como algunos precedentes relacionados con la implementación de aplicaciones en Android. Cada uno de estos permitió rescatar las características de los métodos agiles como las más relevantes a fin de abordar proyectos de este tipo. Una vez entablada la base teórica se evaluaron las características más destacadas según los criterios de sencillez y

eficiencia establecidos para la definición del procedimiento, buscando estructurar de manera organizada aquellas tareas necesarias para la obtención del producto pero que fueran muy intuitivas al momento de su aplicación y que permitieran terminar el proyecto en poco tiempo, con resultados funcionales y con pocas herramientas.

En el capítulo I se definen los conceptos básicos sobre el sistema operativo Android, su arquitectura y las herramientas de desarrollo para este; así como un recuento sobre paradigmas de programación y modelos de desarrollo, con lo cual se establece un conocimiento mínimo necesario para abordar la problemática de la investigación.

En el capítulo II veremos las fuentes sobre desarrollo software para dispositivos móviles actualmente, así como los antecedentes encontrados sobre proyectos que culminaron con la implementación de aplicaciones en Android. Se analizan las metodologías agiles Scrum, XP, TDD y Mobile-D y se evalúan las características relevantes a la definición del nuevo procedimiento. De éste modo se crea una base teórica actualizada que contribuirá a la solución del problema.

En el capítulo III se especifican las fases y tareas definidas para el procedimiento propuesto EfiSenDroid.

En el capítulo IV se detalla la implementación del procedimiento EfiSenDroid al desarrollo del prototipo de la aplicación Assistance Control, la cual permite tomar la asistencia a determinado grupo de estudiantes de modo que se guarde el listado dentro del almacenamiento interno del dispositivo para su posterior revisión o modificación.

PLANTEAMIENTO DEL PROBLEMA

Cualquier tipo de desarrollo software es una tarea compleja que requiere tiempo, análisis, control de etapas, pruebas y gran conocimiento por parte de los directamente implicados. Prueba de ello es que en la actualidad existen numerosas propuestas metodológicas que se enfocan en distintas partes del proceso de desarrollo. En el caso del desarrollo de aplicaciones Android no se tiene ningún modelo o algún estándar establecido que sirva como guía para sus proyectos, debido a que en esencia, las aplicaciones Android son creadas como software común, por lo cual se maneja una mezcla de las metodologías ya establecidas, claro está, con variaciones dependiendo del proyecto que se abarque.

En primer lugar tenemos las metodologías tradicionales que se centran en el proceso, siendo muy rigurosas con las actividades involucradas, en las herramientas usadas y en los artefactos que se deben producir. Por otro lado están las metodologías agiles que tienen como principal dimensión el factor humano o el producto software, dándole mayor importancia al individuo, la colaboración con el cliente y al desarrollo incremental del software. Por esto es necesario conocer las metodologías o procedimientos de desarrollo software móvil existentes, saber cómo funciona cada uno, para así, poder destacar cada una de sus ventajas y debilidades buscando proponer un nuevo procedimiento evaluado por criterios de efectividad y sencillez que permita afrontar proyectos en este campo.

JUSTIFICACIÓN

Debido a que el sistema operativo Android y el desarrollo de sus aplicaciones gozan de gran popularidad en usuarios de dispositivos móviles y que sin embargo para crear Apps de éste tipo no se cuenta con un proceso organizado que permita una correcta implementación, éste proyecto de grado propone un procedimiento para desarrollar aplicaciones propias para Android y que permita abordar proyectos móviles en ésta plataforma.

Desde el punto de vista metodológico, esta investigación genera conocimiento válido y confiable por medio de la recopilación de una base teórica referente a implementación de software móvil basada en proyectos documentados cuyo producto final fue una aplicación Android haciendo uso de métodos propios de ingeniería de software; por otra parte, en cuanto a su alcance, esta investigación aporta un nuevo ciclo de vida estructurado con actividades enfocadas a características propias de Android, ideal para desarrolladores interesados en ésta temática.

Profesionalmente pondrá en manifiesto los conocimientos adquiridos durante la carrera y el desarrollo del proyecto, además permitirá sentar las bases para otros estudios que surjan partiendo de la problemática aquí especificada.

OBJETIVOS

Objetivo General

Elaborar un procedimiento que permita el desarrollo de aplicaciones Android.

Objetivos Específicos

1. Identificar, analizar y evaluar los procedimientos actuales aplicados al desarrollo de aplicaciones de software móvil en Android.
2. Determinar el procedimiento para el desarrollo de aplicaciones con Android.
3. Desarrollar un prototipo operativo que valide el procedimiento.
4. Evaluar la aplicabilidad y la eficiencia del procedimiento en el desarrollo del prototipo.

Acotaciones

- La consulta de las fuentes acerca de desarrollo software se hará a partir de algunos de los procedimientos y metodologías más importantes en la actualidad que hagan uso de la plataforma Android.
- La selección entre las diferentes fuentes consultadas se hará bajo criterios de efectividad y sencillez.
- El proyecto contempla la entrega de una propuesta de procedimiento para el desarrollo de aplicaciones en Android.
- La elaboración de un prototipo operativo de una aplicación sencilla en Android.

- El prototipo será emulado desde un equipo de cómputo portátil y en la medida de lo posible ejecutado desde un dispositivo móvil.

Capitulo I. Marco Teórico

En este capítulo se describen algunos de los temas básicos necesarios para comprender la problemática y el desarrollo de la investigación. La idea es preparar un conocimiento mínimo y aclarar conceptos para aquellas personas que no estén relacionadas con la temática mencionada en el documento. De manera general se abarcan temas como el origen y definición de Android, su evolución y características innovadoras a lo largo de las diferentes versiones. Se hace referencia a los paradigmas de programación actuales como enfoques para diseñar soluciones a diferentes problemas debidamente delimitados, también se detallan las actividades comunes y modelos de desarrollo estándar para software. Finalmente se hace mención al kit para implementación propio de la plataforma Android y algunos de los entornos de desarrollo integrados oficiales de Google que se tendrán en cuenta en este proyecto.

1.1. ANDROID

Desde sus inicios fue concebido como un sistema operativo, desarrollado por primera vez por Android Inc. (2003) a manos de Andy Rubin, basándose en el núcleo de Linux y cuyo diseño se aplicaría principalmente a dispositivos móviles como Smartphone y Tablet con pantallas táctiles. En 2005, después de 2 años de fundada, la empresa de Android Inc. fue adquirida por Google, donde como era de esperarse, Andy pasó a formar parte y donde tiempo después se convertiría en vicepresidente de ingeniería manteniéndose directamente involucrado en el desarrollo de Android. Android fue presentado dos años después, el 5 de noviembre de 2007 de la mano de la Open Handset Alliance (OHA), la cual es un conglomerado de 84 compañías dedicadas al desarrollo de estándares para dispositivos móviles, consta con miembros como Google, HTC, Dell, Motorola (Moto -Lenovo), Qualcomm, Samsung,

LG y Nvidia. Sin embargo fue solo hasta el 22 de octubre del 2008 que estuvo a la venta el primer Smartphone que ya corría Android versión 1.0, este fue el caso del HTC Dream (Hebuterne y Pérochon, 2014).

Aunque originalmente se pensaba implantarlo exclusivamente en móviles tuvo gran aceptación dentro del mercado por lo cual actualmente funciona en ordenadores, portátiles, Google TV, relojes de pulsera, auriculares, vehículos, lavadoras y otros dispositivos. La plataforma Android es de carácter abierto, originalmente con licencia GNU GPLv2 para el kernel y la Apache v2 para los otros componentes del sistema. Las aplicaciones desarrolladas para Android se codifican en lenguaje Java pero con unas API's propias de modo que los códigos normales de Java no sean compatibles con este sistema.

1.2. ARQUITECTURA ANDROID

Android está basado en software libre y se compone de cuatro capas bien diferenciadas tal como se observa en la Figura 1. De esta forma los desarrolladores pueden construir aplicaciones completamente funcionales aprovechando el hardware del dispositivo de la mejor manera, los servicios en segundo plano, agregar notificaciones, etc.

A través del núcleo de Linux se proporcionan servicios esenciales para seguridad, manejo de memoria, multiprocesos, y soporte de controladores. Debido a las limitaciones en hardware de los dispositivos móviles fue necesario implementar una nueva máquina virtual denominada Dalvik, ya que no era posible el uso de una estándar de java. En las últimas versiones de Android la máquina virtual se ha reemplazado por ART en busca de un mejor desempeño y reducciones de tiempo.

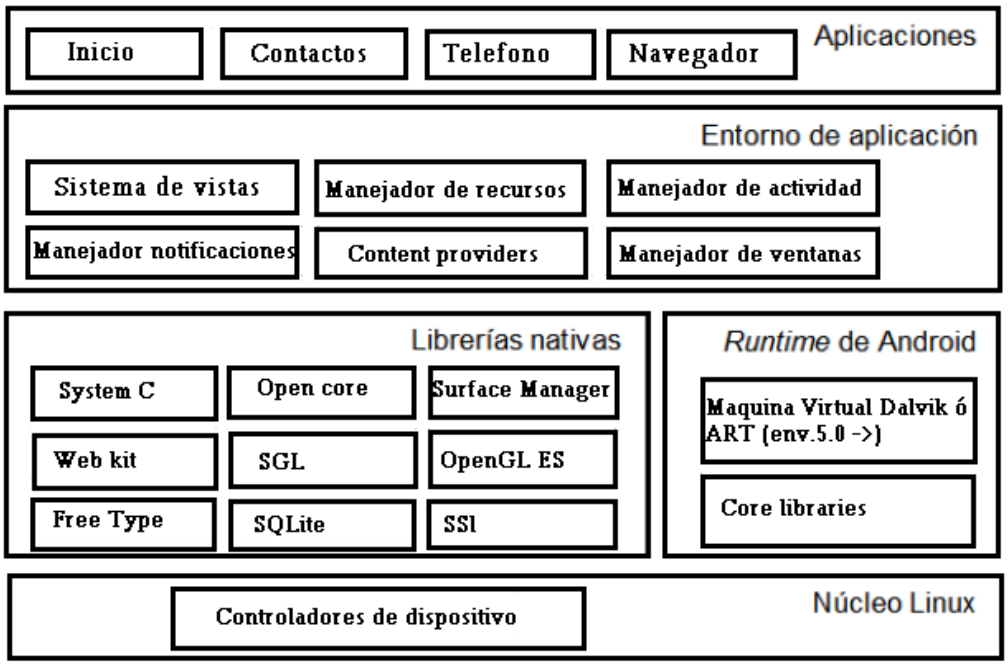

Figura 1. Arquitectura del sistema operativo Android

Fuente: Adaptado de *Arquitectura Android*. (s.f) Recuperado de https://sites.google.com

Incluye numerosas librerías C/C++ para equipos basados en Linux, la librería Media Framework con soporte para codecs de audio y video, Chromium que es la base del navegador Chrome, librerías 3D, SGL para proyección de gráficos en 2D y SQLite como ligero motor de bases de datos.

En el Entorno de Aplicación se encuentra la plataforma necesaria para impulsar el desarrollo y funcionalidad de cualquier tipo de aplicación, permite la reutilización y el reemplazo de componentes mientras se encuentren dentro de los parámetros de seguridad. Los desarrolladores pueden acceder a servicios como Resource Manager para acceder recursos diferentes a código, Activity Manager para el ciclo de vida de las aplicaciones y como se navega entre estas, Notification Manager en caso de que la aplicación deba mostrar alertas en la barra de notificaciones y Content Providers si es necesario acceder a datos de otras aplicaciones.

1.3. VERSIONES DEL SISTEMA

Desde el lanzamiento de Android en 2007 con su primera versión dentro del mercado se han venido desarrollando una serie de actualizaciones posteriores a este sistema operativo. Dichas actualizaciones se han presentado bajo nombres clave de orden alfabético, las versiones hasta el día de hoy son (La historia de Android, 2014):

1.3.1. Android 1.0 Apple Pie – Angel Cake (Alpha)

Primera versión estable del OS, en español “Tarta de manzana”. Fue lanzado por primera vez el 23 de septiembre del 2008, incorporaba las características más bajas que se podían esperar en un Smartphone, entre las cuales están: Android Market (hoy en día Google Play), Gmail, Google Maps, Talk, entre otras. No contaba con teclado virtual, se hacía uso del teclado físico, además no tenía soporte para HTML5 ni para el flash de la cámara.

1.3.2. Android 1.1 Banana Bread - Battenberg (Beta)

La actualización salió a la luz el 9 de febrero de 2009, conocida en español como “Pan de plátano”, a diferencia de la primera versión en esta se resolvieron algunos fallos del sistema, y se realizaron cambios en su interface gráfica además de otras funcionalidades no tan relevantes.

1.3.3. Android 1.5 Cupcake

En la actualidad no quedan modelos que corran esta versión. Una de las particularidades más llamativas es la inclusión de un teclado virtual, el cual además soporta la funcionalidad de predicción de texto y diccionario. Aparecen por primera

vez los Widgets, carpetas y accesos directos en la interfaz principal además de una serie de mejoras multimedia en la grabación de audio y video con soporte bluetooth.

1.3.4. Android 1.6 Donut

En esta nueva entrega se incorpora la funcionalidad para captación de gestos así como la síntesis texto/voz que permite a las aplicaciones poder “leer” algún tipo de texto. Otro aporte es la mejora en los métodos de búsqueda y búsqueda avanzada a lo largo del dispositivo tanto en el ámbito textual como por medio de voz. Además Android Market fue rediseñado para mostrar aplicaciones gratis como pagas por medio de un catálogo, haciendo más fácil la búsqueda de estas.

1.3.5. Android 2.0/2.1 Éclair

Con Éclair las características de Android se fueron afianzando, esta vez aparecen cambios relevantes para optimizar el rendimiento por medio de aceleración en el hardware, se amplía el soporte para pantallas más grandes así como mejoras de resolución, esto permite a su vez la mejora en el manejo del teclado virtual y de las aplicaciones de navegación.

1.3.6. Android 2.2 Froyo

Aun hoy en día se puede encontrar uno que otro equipo corriendo esta versión, su principal atracción fue el rediseño de su pantalla, se agregan 2 paneles más en la pantalla de inicio con respecto a los 3 que se tenían en anteriores versiones y se agregó la figura de puntos la cual sirve como guía para saber la ubicación actual.
Por otra parte, aparece una nueva funcionalidad que permite crear un punto de acceso a Wifi desde el dispositivo por medio de la conexión 3G de este, por medio del cual otros equipos se pueden conectar y navegar en internet.

1.3.7. Android 2.3 Gingerbread

Actualmente esta es la versión mínima que se puede encontrar en algún dispositivo con Android, es muy difícil encontrar equipos que aun corran Froyo o sus antecesores. A partir de esta versión aparecen mejoras significativas enfocadas en el aumento del tamaño de la pantalla y una mayor resolución, además se corrigen fallos en la función que permite el copiado, cortado y pegado dentro del dispositivo. Quizá una de las características más llamativa sea la inclusión de cámara frontal así como el soporte para la misma, esto pensando en las aplicaciones que permiten la opción de videoconferencia.

1.3.8. Android 3.0 Honeycomb

Esta versión es una de las menos conocidas, esto se debe a que fue liberada únicamente para dispositivos Tablet, a partir de esta actualización se ven reflejadas muchas de las mejoras en rendimiento que posteriormente serían ampliadas. Una de estas mejoras es la que permite por primera vez soportar procesadores multinúcleo, permitiendo mayor velocidad en el despliegue de varias aplicaciones simultáneamente. Aparecen nuevas opciones dentro del menú de notificaciones que permiten acceso rápido a diferentes funcionalidades del sistema así como la posibilidad de ver las múltiples tareas que se están ejecutando y cambiar entre éstas rápidamente.

1.3.9. Android 4.0 Ice Cream Sandwich

Una de las versiones más atractivas entonces y de la cual aún disponen muchos equipos hoy en día. El principal de sus atractivos es la posibilidad de funcionar casi por completo sin la necesidad de botones físicos, equipados para dispositivos móviles completamente táctiles y con cambios en la presentación de su interfaz de inicio.

Se hacen visible cambios para mejorar la privacidad de los dispositivos, la pantalla de bloqueo incorpora diferentes tipos de seguridad para acceder a las funcionalidades del equipo. Se centra especial atención en la forma como se crean carpetas simplemente arrastrando y agrupando accesos de las aplicaciones, así como en las opciones integradas dentro de la aplicación de cámara.

1.3.10. Android 4.1-4.3 Jelly Bean

Esta es la versión más estable y de mejor rendimiento de las lanzadas hasta este punto, al igual que sus antecesores, esta incluye corrección de errores en funcionalidades ya existentes, mejoras en el browser de navegación y en las aplicaciones multimedia para descarga, reproducción y grabación de audio y video, mayor personalización del fondo con imágenes animadas, así como introducción de texto por dictado y el desbloqueo por medio de reconocimiento facial.

Entre las nuevas particularidades encontramos la inclusión de nuevos idiomas y el soporte Bluetooth, además de poner especial atención en las opciones de ahorro de batería, la cual en casi todos los dispositivos móviles es la mayor de sus desventajas.

1.3.11. Android 4.4 Kitkat

El mayor de sus atractivos es la posibilidad del sistema para poder ejecutarse en muchos dispositivos sin importar la gama, esto debido a la disminución en sus exigencias de funcionamiento, como la memoria RAM, permitiendo su ejecución en dispositivos de tan solo 512MB. Desde su pantalla de inicio se pueden observar cambios importantes y de mayor atractivo visual, entre estos están el incremento en el tamaño de los iconos y el texto que ahora se condensa, la barra de estado, que aparece en la parte superior, ahora es transparente permitiendo una mezcla con el

fondo de pantalla. También aparece la reproducción de música en segundo plano sin la necesidad de desbloquear el equipo.

1.3.12. Android 5.0/5.0.1/5.0.2/5.1/5.1.1 Lollipop

Esta es la penúltima versión del OS Android, fue dada a conocer el 25 de julio de 2014 pero su primera aparición fue hasta el 3 de noviembre del mismo año. En primera instancia fue liberada la versión 5.0, la cual después de ser adaptada por cada fabricante comenzaría su despliegue a los dispositivos. Los primeros en recibirla fueron equipos de alta gama como la línea Nexus y las nuevas versiones Motorola de finales del 2014, posteriormente otras marcas dentro de los Estados Unidos y después el resto del mundo.

Uno de los principales inconvenientes con esta versión fueron los múltiples bugs encontrados, que hacían del sistema algo “indeseable”. Como corrección de esto Google se comprometió a dar rápida solución por lo que a los equipos que ya habían recibido Lollipop se les libero una nueva actualización del sistema denominada 5.0.1 y 5.0.2, en las cuales se corregían algunos de estos errores.
Sin embargo a finales del mes de Abril se dio a conocer que una nueva versión estaría empezando su despliegue, en este caso hablamos de Lollipop 5.1 - 5.1.1 en las cuales se incluyen nuevas correcciones del sistema operativo.

1.3.13. Android 6.0 Marshmallow

Tras los tropiezos presentados con Lollipop Google lanzó la última versión de Android denominada “Malvavisco” en español, la cual promete ser una de las versiones más estables y que podría tener muy buena acogida dentro del público en general. En esta ocasión hay un enfoque especial en cuanto al manejo de la batería con la inclusión de funciones como Doze, el cual es un estado parcial de suspensión automático cuando no se está haciendo uso de dispositivo, App Standby

por su parte se ocupa de aquellas aplicaciones que se ejecutan en segundo plano haciendo un uso innecesario de batería. En términos de seguridad hay cambios en relación a los permisos que se otorgaban a las aplicaciones una vez que iban a ser instaladas, con Marshmallow los permisos pueden ser denegados sin que impida el uso de dicha aplicación. En el caso de algunos dispositivos de gama media y los de alta dispondrán de la función para desbloque por medio del sensor para huella digital, así como para confirmaciones en los usos de tarjeta.

Se destacan también funciones que ya venían con Lollipop pero que son remasterizadas en esta nueva entrega y otras nuevas como la opción de eliminar las capturas de pantalla desde la barra de estado, nueva funcionalidad para la gestión de la memoria RAM, inclusión del modo "No Molestar" que deja en total silencio el dispositivo, nuevas opciones dentro del menú Recovery y una nueva visualización en el modo como se contesta el teléfono.

1.3.14. Android 7.0 Nougat

La última versión liberada en la actualidad trae consigo varios cambios importantes tanto en el diseño con en varias de las funciones del sistema. Aunque en total se incluyen más de 200 novedades se destacan solo algunas de uso cotidiano del usuario común.

Uno de sus principales atractivos es la inclusión del modo multiventana de manera nativa en el sistema (véase Figura 2), lo cual permite la ejecución simultanea de dos aplicaciones en la misma pantalla con la posibilidad de modificar sus tamaños tal cual como lo haríamos en un computador. Aunque esta funcionalidad es bastante llamativa, cabe resaltar que no todas las aplicaciones son compatibles para su uso hasta el momento y que sus mejoras dependerán del aprendizaje capturado por sus desarrolladores para futuras versiones.

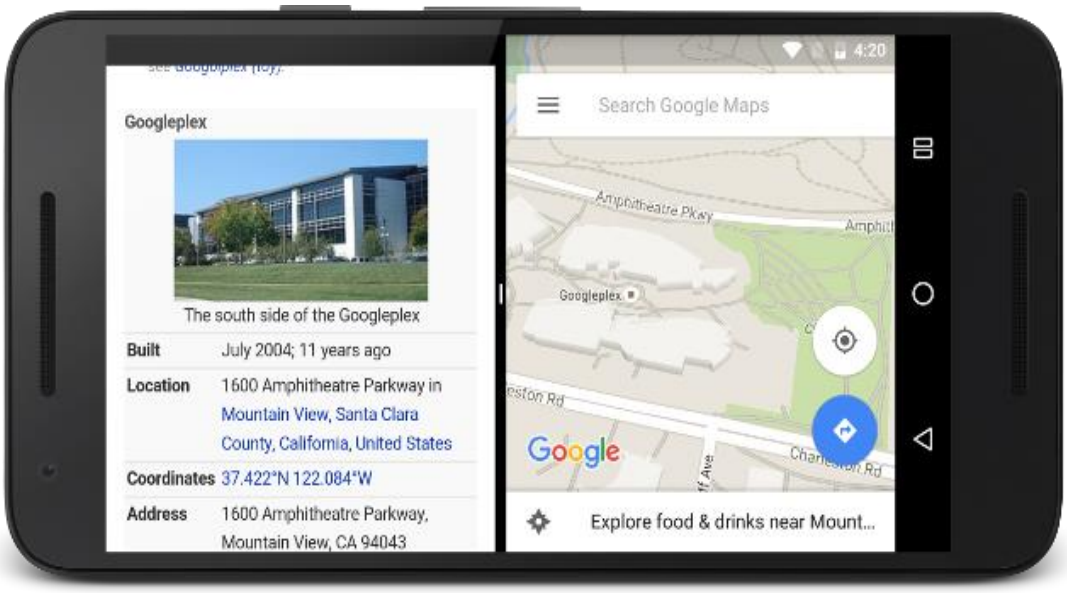

Figura 2. Modo multi-ventana nativo en Android 7.0 Nougat.

Fuente: Nickname: Cosmos (2016). Así es la pantalla dividida de Android N [Imagen]. Xataka Android. Recuperado de http://www.xatakandroid.com

1.4. PARADIGMAS DE LENGUAJES DE PROGRAMACION

Hoy en día, existen un sinnúmero de lenguajes de programación, cada uno con formas y propósitos distintos los cuales pueden ser clasificados en dos grandes grupos en los que se encierran casi todos los paradigmas que se conocen hasta el presente. Estos dos grupos que se muestran en la Figura 3 son la Programación Declarativa: funcional, lógica y la Programación Imperativa: orientada a objetos, visual, orientada a eventos, orientada a aspectos, etc.

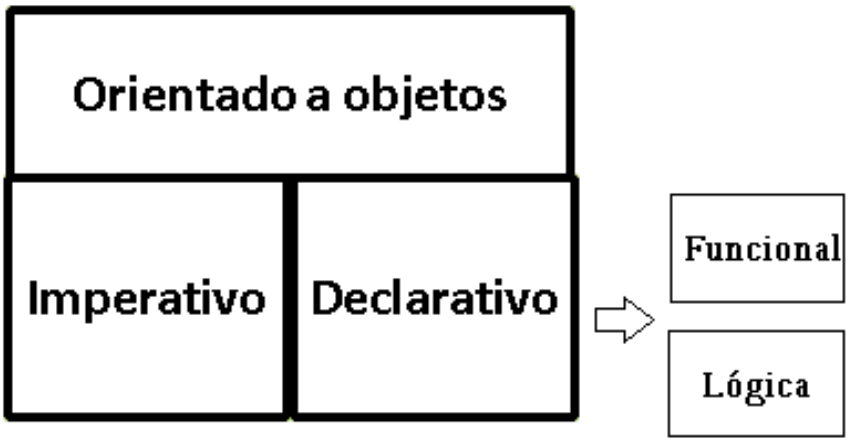

Figura 3. Clasificación General de los Paradigmas de Programación

Fuente: Zárate, H. (2008), *Paradigmas de programación*, Ciudad de México, México: Creative Commons.

1.4.1. Programación Funcional

El paradigma funcional está basado en el concepto matemático de función: "Una función *f* asigna a cada miembro de un conjunto *X*, exactamente un miembro de un conjunto *Y*. Dónde: E conjunto *X* y *Y* puede o no ser el mismo y donde X es llamado *dominio de f* y Y es llamado *co-dominio o rango de f*" (Joyanes, 2008).

El programa es considerado por el paradigma funcional como una función matemática, donde el dominio representa todas las entradas (inputs) y el rango representa todas las salidas (outputs). Según se observa en la Figura 4 se puede decir que cualquier clase de programación se puede catalogar como programación funcional y esto es relativamente cierto exceptuando los siguientes puntos clave:

Figura 4. Las entradas, se operan mediante una definición y se obtiene una salida

Fuente: Zárate, H. (2008), *Paradigmas de programación*, Ciudad de México, México: Creative Commons.

- Primero, que en los lenguajes funcionales no existe el concepto de variables y consecuentemente tampoco existen operaciones de asignación. Aunque podría parecer que los parámetros de una función se comportan como una variable (puesto que pueden llevar un nombre y su uso es de hecho parecido al de las variables), esto es incorrecto: El valor de los parámetros es dado como fijo al inicio de la función.
- Segundo, la existencia de una propiedad llamada transparencia referencial (Referential transparency), la cual indica que una función sólo depende de sus parámetros y que tendrá efecto únicamente en su resultado, por lo que podríamos llamar a una función arbitrariamente sin tener efectos colaterales en el resto de las computaciones.
- Por último, que una función puede ser utilizada como parámetros y resultados de cualquier otra función.

1.4.2. Programación Lógica

Tanto la lógica como la sintaxis de este paradigma representan conocimiento a través de inferencias. Trabajar con este paradigma especifica que hacer y no como hacerlo, es por esto que son llamados lenguajes declarativos.

El proceso general de la programación lógica es que a partir de un conjunto de reglas e inferencias podamos comprobar nuevas proposiciones que no sean relevantes. Este proceso está basado en reglas de lógica de primer orden (Zárate, 2008 p.6).

1.4.3. Programación Orientada a Objetos

El paradigma de la programación orientada a objetos hace un cambio en el desarrollo de software, implantando un nivel de abstracción alto, permitiendo

mejorar el código final. De manera básica, según exponen Rodríguez, Prieto y Sosa (2011), las aportaciones de este paradigma se pueden resumir en:

- Conceptos de clase y objeto, que proporcionan una abstracción del mundo centrada en los seres y no en los verbos.
- Los datos aparecen encapsulados dentro del concepto de clase. El acceso a los datos se produce de manera controlada e independiente de la representación final de los mismos. Como consecuencia, se facilita el mantenimiento y la evolución de los sistemas, al desaparecer las dependencias entre distintas partes del sistema.
- Mediante conceptos como la composición, herencia y polimorfismo se consigue simplificar el desarrollo de sistemas. La composición y la herencia nos permiten construir clases a partir de otras clases, aumentando en gran medida la reutilización.

Este paradigma hace posible el desarrollo modular de un software, porque, hasta cierto grado, cada componente es independiente de los demás, lo que permite que estos componentes puedan ser reutilizados y reciclados, incluso, a lo largo de distintos proyectos. Es posible que esta “modularidad” haya colocado a la programación orienta-da a objetos en la posición dominante en la que se encuentra ahora, dado que le ha permitido a la industria encontrar nuevas formas de trabajo y patrones de diseño más productivos (Rodríguez et al., 2011).

Gran parte de los lenguajes más utilizados hoy en día están orientados a objetos, entre ellos se encuentran: Java, C++, PHP, Python, C#, Delphi, Ruby, D y ActionScript, entre otros.

1.5. PROCEDIMIENTO DE DESARROLLO SOFTWARE

Un procedimiento puede verse como una estructura aplicada al desarrollo, la planificación y el control de algún tipo de producto software. Esta describe una secuencia de actividades que deben ser seguidas por el grupo de trabajo de modo que se pueda cumplir con las fechas establecidas. El objetivo básico dentro del proceso es el de poder predecir el trabajo que se requiere, es decir, el costo, nivel de calidad y no menos importante el tiempo de desarrollo. Actualmente no existe un proceso para desarrollar software universal, debe configurarse de acuerdo con la naturaleza del producto y de las necesidades de la empresa, además de las herramientas a utilizar así como las entregas o versiones requeridas y los controles, permitiéndole crear un producto con costo predecible y alta calidad (Jacaboson, Booch, y Rumbaugh 2000).

Los procedimientos son alterados, en su mayoría, por factores intelectuales, creativos y por juicio de los desarrolladores. Cada una de estas variaciones depende del nivel de complejidad del producto software que se busque obtener, el cual en estos casos es intangible y en forma general muy abstracto lo que impide su temprana definición. En la Figura 5 se observa de manera general como los requerimientos necesarios para cualquier proyecto son recogidos por alguno de los diferentes procedimientos existentes y una vez atravesado el ciclo de vida se tendrá como producto un nuevo sistema según los requisitos procesados.

Figura 5. Procedimiento de desarrollo software

Fuente: Elaboración propia.

1.6. ACTIVIDADES GENERALES DEL DESARROLLO DE SOFTWARE

Desde hace ya mucho tiempo se viene hablando acerca del proceso que sigue un software como “ciclo de vida”, esto es, desde que empieza a establecerse como proyecto hasta que no es más utilizado, pasando por una serie de etapas consecutivas relacionadas con el estudio de su origen, las funcionalidades, el tratamiento de sus restricciones, establecer su diseño, desarrollarlo, realización de pruebas, implantación, uso y realización de tareas de mantenimiento (IEEE).

Aunque para cada caso se deban tomar medidas diferentes de acuerdo a la índole del proyecto, existen unas pautas que permiten organizar el proceso, estas son actividades que se repiten una y otra vez y de las cuales se encuentran en forma general unas que hacen parte de la mayoría de los procedimientos:

1.6.1. Planificación

Esta etapa, en general, sin importar el tipo de procedimiento que se esté utilizando es de vital importancia debido a la obtención de requisitos así como el análisis de los mismos. Su importancia radica en que en la mayoría de los casos la idea de los clientes suele ser abstracta según el resultado final esperado, aunque tienen claras las funciones que debe desempeñar el software. A partir de esta recopilación de requerimientos del cliente se puede proceder a analizar el ámbito respectivo al proceso de desarrollo.

1.6.2. Implementación, pruebas y documentación

Es esta parte es donde el equipo de desarrollo procede con la elaboración del código para el proyecto. Una vez que el código es parcialmente terminado se inician las pruebas al código puesto que es vital la detección de errores en el software antes de que pase a la siguiente etapa.

En el transcurso de estas y las demás etapas del proyecto se va estableciendo la documentación del diseño interno del software, la cual más adelante servirá como fundamentación para facilitar la mejora y mantenimiento del producto.

1.6.3. Despliegue y mantenimiento

Cuando se han hecho pruebas suficientes al código se procede a liberarlo y distribuirlo para su producción. En esta sección del proceso se debe continuar con la capacitación del personal que dará uso al producto, de modo que este sea utilizado de forma amena para los clientes finales. Además se deberá estar al pendiente del software que se entrega de modo que se puedan encontrar posibles fallos o problemas o la necesidad de la inclusión de nuevas funcionalidades.

1.7. MODELOS DE DESARROLLO

Cuando se habla de modelos de desarrollo de software se pueden entender como una representación abstracta la cual no define como se debe desarrollar sino que refleja un enfoque común. En la mayoría de los casos representan una secuencia entre las actividades, objetos, modificaciones y eventos que plasman las estrategias para lograr la evolución del software. Existen varios modelos los cuales cuentan con ventajas y desventajas; para abarcar determinado proyecto siempre es necesario realizar un buen análisis del problema de modo que se aplique el modelo más apropiado a las necesidades de dicho problema, cabe resaltar que en muchas ocasiones es necesaria la combinación de varios de estos modelos. Actualmente existen tres paradigmas de desarrollo de software (Sametinger, 1997):

1.7.1 Paradigma Tradicional

Es uno de los paradigmas más antiguos, esencialmente consiste en dividir un problema en partes más pequeñas de modo que sea más accesible y manejable.

Como todo modelo, permite identificar ventajas y desventajas de acuerdo a la fundamentación recogida en los proyectos en los que fueron aplicados, la Tabla 1 muestra de manera general algunas de las características fuertes en este tipo de paradigma, así como sus debilidades.

Tabla 1
Ventajas y desventajas Paradigma Tradicional

Ventajas	Desventajas
• Mayor control en cuanto a la programación del desarrollo • Al tener control, se reduce el riesgo de exceso de gastos	• El usuario no participa en el proceso de desarrollo • El proceso no se hace de forma secuencial • El tiempo de desarrollo excede al estimado • Si el usuario olvida aclarar pautas, esto puede significar sobrecostos en el proyecto

Nota: Adaptado de *Cuadro de ventajas y desventajas del uso del Paradigma tradicional*, (2014). Wikipedia. Recuperado de https://es.wikipedia.org/

La aplicación de este tipo de paradigma conlleva la creación de partes más pequeñas del problema principal, esto con el fin de poder resolver de manera sencilla las partes más pequeñas y así ir hallando una solución de forma estructurada que permita el resultado del problema general.

1.7.2 Paradigma Orientado a Objetos

Como su nombre lo indica, este paradigma está basado en la programación con objetos, la cual hace uso de estos mismos y de sus interacciones para diseñar diferentes tipos de Apps, unas de sus principales características pueden describirse de la siguiente forma:

- Posibilita la re-utilización de software.
- Permite el desarrollo de herramientas informáticas por medio de una simple implementación con notación UML.

Los objetos son entidades que tienen un determinado estado, comportamiento (método) e identidad:

- La composición del estado es puramente datos o información en sus atributos.
- El comportamiento hace referencia a aquellos métodos u operaciones que dicho objeto maneja para dar un resultado.
- Su identidad es básicamente la propiedad que tiene cada uno de ser identificado entre los demás objetos.

En la Figura 6 vemos como un objeto individual posee toda la información que permite identificarlo así como los métodos que le permiten la comunicación e interacción entre objetos. La programación orientada a objetos se diferencia de la tradicional en su modalidad operacional, en la estructura tradicional primero se definen términos de procedimientos y después las estructuras de datos que dichos procesos deberán manejar; por el contrario la POO define en primera instancia objetos y después envía mensajes para que estos realicen dichos métodos por si solos.

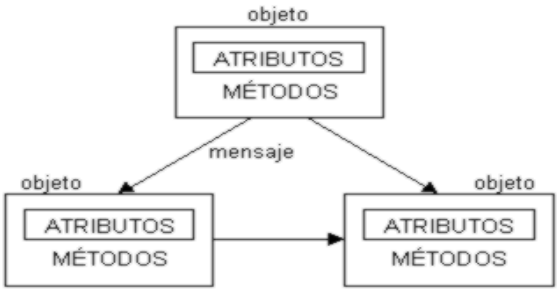

Figura 6. Estructura de un objeto

Fuente: Elaboración propia.

Usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento. En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos.

1.7.3 Paradigma de desarrollo Ágil

Este tipo de paradigma, como su nombre lo indica, se basa en la modelación de procesos agiles. En sus inicios este tipo de metodología era conocida como livianas puesto que evitaban el tortuoso estilo del paradigma tradicional.

Usa un enfoque sobre la construcción del software de la mano del cliente, promoviendo la inclusión de diferentes cambios dentro de la continuidad del proceso. En este tipo de paradigma se desarrolla a través de una serie de iteraciones las cuales tienen como objetivo una pequeña entrega del producto entre una iteración y otra, en general cada una debería tardar aproximadamente unas cuatro semanas.

Una de las principales características de los métodos agiles es que después de cada iteración el equipo re-evalúa cada una de las prioridades del desarrollo haciendo que la siguiente sea más consistente con lo que se quiere entregar al cliente.

1.8 ANDROID SDK (KIT DE DESARROLLO SOFTWARE)

El Kit de desarrollo para Android, al igual que cualquier otro SDK, hace referencia al conjunto de herramientas que permiten la creación de software aplicativo para un sistema concreto, en este caso el SDK de Android comprende una bien elaborada biblioteca, el depurador y un simulador basado en cada una de las versiones de Android disponibles. Cada aplicación elaborada contiene empaquetados los ficheros correspondientes en formato .apk el cual incluye ejecutables de Dalvik.

Este entorno puede ser tan sencillo como una simple interfaz de programación (API), la cual permite el uso de lenguajes de programación, en este caso Java y XML. La plataforma integral oficial era Eclipse haciendo uso del plugin ADT, también se encuentran disponibles otras extensiones que permiten desarrollar de igual forma dentro del IDE NetBeans.

1.8.1 Android Studio 2.1

A partir de 2014 Google anuncio oficialmente Android Studio como su nuevo entorno de desarrollo integral para la creación de aplicaciones Android. Basado en licencia Apache 2.0 se publicó de manera gratuita para estar al alcance de todos los desarrolladores, es una herramienta multiplataforma que facilita procesos para edición, depuración y compilación de código. Gracias a Instant Run se acelera la edición y ejecución haciendo notablemente más rápida la creación de los bloques de código que junto con la opción de emulador permite visualizar los cambios al instante. Su entorno unificado permite el desarrollo de aplicaciones para diferentes tipos de dispositivos.

1.8.2 App Inventor 2 (Beta)

Desarrollado por la unidad de Google Labs, la cual servía como zona de prueba de diferentes proyectos que aún no iban a salir al mercado. Es una plataforma que permite el desarrollo de aplicaciones para Android sin la complejidad de la edición de las líneas de código. App Inventor permite desarrollar, a partir del uso de la librería Open Block de Java, diferentes tipos de aplicaciones de manera mucho más sencilla e intuitiva, además permite visualizar en tiempo real cada uno de los cambios que se van haciendo.

Capitulo II: Estado del Arte

Los métodos actuales para desarrollo poseen un enfoque completamente disciplinado que permite la elaboración de productos con altos márgenes de portabilidad y fiabilidad, todo esto gracias al papel de la ingeniería del software en cada uno de ellos. En el caso del desarrollo de aplicaciones móviles no se ha establecido un modelo que abarque cada una de las características especiales para este tipo de implementaciones, se aplican los mismos métodos estructurados para software común. El único referente al respecto es acerca del uso de métodos ágiles, por lo cual se seleccionan los procedimientos de este tipo que mejor referenciados se encuentran actualmente y de los cuales se encuentra mejor soporte o documentación en la actualidad.

Una de las características importantes de la gran mayoría de los desarrollos móviles es su corta duración. Esto se debe a factores como la gran competencia en el sector, los cambios en el mismo con la aparición, casi constante, de novedades tanto en software como hardware, el hecho de que muchas aplicaciones nacen con un desarrollo precoz o prototipo (y van evolucionado después) o incluso la simplicidad de las aplicaciones que no requieren grandes desarrollos. Esta suele ser, salvo algunas excepciones, la norma de los desarrollos de aplicaciones para dispositivos móviles.

2.1. Desarrollo Ágil

Los procedimientos o enfoques de tipo ágil difieren de las metodologías debido a que en su esencia son mucho más ligeras, es decir, no son tan complejas o pesadas. Aunque se habla de este tipo de enfoques desde mediados de los 80’s apenas un par de años atrás han tomado verdadera importancia dentro de la

ingeniería del software, en éste caso se habla particularmente del desarrollo de aplicaciones móviles.

Una de las principales características que trae consigo la aplicación de estos procedimientos es el cambio en la forma de pensar tanto organizacionalmente como por equipos, se deja de lado la ideología de seguir siempre un proceso, un método, una estructura definida para lograr un buen funcionamiento, esto debido a que en la actualidad existe una tendencia al cambio en indeterminada cantidad de aspectos, por lo cual es necesaria la adaptación constante. Aún más importante, es el hecho de tener a las personas como centro de este tipo de enfoque, es decir, priman los individuos y sus interacciones sobre los procesos. Como podemos ver en la Tabla 2, el esfuerzo más importante recae sobre la creatividad de las personas con talento, estos modelos son difícilmente panificables, siempre se está atento a la posibilidad de incluir alguna modificación, además demuestran como las pruebas y la revisión constante del código permite productos de calidad.

Tabla 2.

Principales diferencias entre ágil y tradicional

Agiles	Tradicionales
• Preparadas al cambio, modificar software es barato • Se enfatiza una constante interacción-comunicación entre los miembros • Proceso menos rígido, pocos principios, menor documentación • No hay contrato tradicional, mayor flexibilidad	• Existe una resistencia al cambio o las modificaciones, es costoso • El proceso y las herramientas priman sobre las interacciones • Proceso altamente controlado, numerosas políticas/normas, gran preocupación por la documentación • Existe un contrato pre-establecido

• Presencia del cliente, hace parte del equipo de desarrollo • Equipos pequeños (<10 personas), se trabaja en un mismo sitio sin secciones • Pocos artefactos • Pocos roles • Mucho menos énfasis en la arquitectura del software • Producción de código con valor	• El cliente tiene muchas menos comunicación con el equipo de desarrollo • Equipos grandes, distribuidos por etapas del proceso • Mas artefactos • Mas roles • La arquitectura es esencial y se expresa mediante modelos • Entorno de desarrollo

Nota: Adaptado de José, H. et al (2004). *Metodologías agiles en el desarrollo de software.*

Se contemplan cuatro postulados que derivan de los procedimientos agiles y los cuales se encuentran en el manifestó original (Beck et al., 2001).

- La interacción con los individuos por encima de los procesos y las herramientas.
- Software funcional sobre exceso de documentación.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta al cambio antes que seguir un plan.

En estos postulados se simplifica toda la filosofía del enfoque ágil, y es a partir de estos que nacen los conceptos que lo definen. Se habla entonces de suavizar los procesos de las metodologías tradicionales integrando cambios como la visualización de flujo de trabajo, mejoras continuas, retroalimentación o feedback continuo y el trabajo en equipo para lograr entregas de valor para el cliente.

La construcción de software es muy diferente a construir, por ejemplo, una vivienda, la cual por lo general antes de iniciar ya se tienen listos los planos, los cuales muy

pocas veces son modificados y son bastante precisos gracias a la ayuda de las matemáticas, por el contrario, con el software no es posible separar con claridad la fase del diseño de la construcción. En la Tabla 3 se hace un análisis comparativo sobre algunas propiedades que hacen a los procedimientos agiles totalmente aplicables al dominio software en móviles.

Tabla 3.

Comparativa entre propiedades agiles y desarrollo móvil

Propiedades Agiles	Motivación	Desarrollo móvil
Volatilidad del entorno	Debido al cambio habrá menos necesidad de diseño y planificación inicial y mayor desarrollo e iteraciones	Gran incertidumbre, entornos dinámicos, nuevos dispositivos constantemente
Equipos de drllo pequeños	Reacción rápida, intercomunicados, se comparte la información, poca documentación	La mayoría de los proyectos son para microempresas, Pymes o particulares
Cliente identificable	No hay malentendidos, asesora el desarrollo de principio a fin	Gran número de usuario finales pero el cliente es fácil de identificar
Entorno orientado a objetos	Las herramientas actuales se orientan a objetos	Usan lenguajes como Java y C++ los cuales también trabajan con objetos
Software critico	Los fallos no llegan a causar pérdidas de vida	Software para entretenimiento, inofensivo
Sistemas pequeños	Menos necesidad de diseño preliminar	Las apps varían en tamaño pero no suelen llevar cientos de miles de líneas de código
Ciclos cortos de desarrollo	Se retroalimentan rápidamente	Periodos de desarrollo pequeños, de 1 a 6 meses

Nota: Adaptado de Rahimian, V., Ramsin, R. *(2008) Designing and agile methodology for mobile software development: a hybrid method engineering approach.* Second International Conference on Research Challenges in Information Scienc *(p 337-342). Marrakech.*

2.2. Scrum

Dentro del amplio mundo del desarrollo ágil, es el marco de trabajo más usado actualmente. Se basa principalmente en desarrollar problemas adaptativos trabajando en equipo para la entrega de productos con el mayor valor posible.
En éste, se realizan entregas parciales del producto en proyectos donde son necesarios los resultados pronto, éstas son supervisadas directamente por el cliente permitiendo así la inclusión de nuevos cambios, revisión de la calidad o evitar que se alargue demasiado. Scrum ha sido implementado por cientos de empresas en miles de proyectos, ya que tiene la confianza no solo de la industria del software sino de industrias de manufactura, innovación y en general por sectores que requieren una gran flexibilidad en la creación de sus productos y contenido (Schwaber, 2016). Los roles principales en Scrum son:

- **El equipo Scrum**
    - Cliente (Product Owner)
    - Facilitador (Scrum Master)
    - Equipo (Development Team)

El cliente es la representación del dueño y usuario final del producto, es la persona de la cual se desprenden las características sobre que construir y en qué orden por medio de historias; el facilitador es quien guía al equipo de trabajo para el cumplimiento de las tareas siguiendo la metodología, junto con el cliente trabajan para reducir los impedimentos; el equipo de desarrollo representa a cada uno de los programadores involucrados, los cuales poseen los conocimientos necesarios para trabajar en el proyecto. EL proceso SCRUM, como se observa en la Figura 7, es ejecutado en lapsos de tiempo cortos y fijos (iteraciones de aproximadamente cuatro semanas) los cuales son denominados *Sprint,* cada uno de estos debe proporcionar como resultado un producto terminado, el cual será un incremento del producto final

y que podrá ser entregado al cliente para su revisión en cualquier momento. Todo esto parte de una lista (Product *Backlog*) que contiene los requisitos del producto a desarrollar, la cual es creada por el cliente y funciona como plan para desarrollar el proyecto.

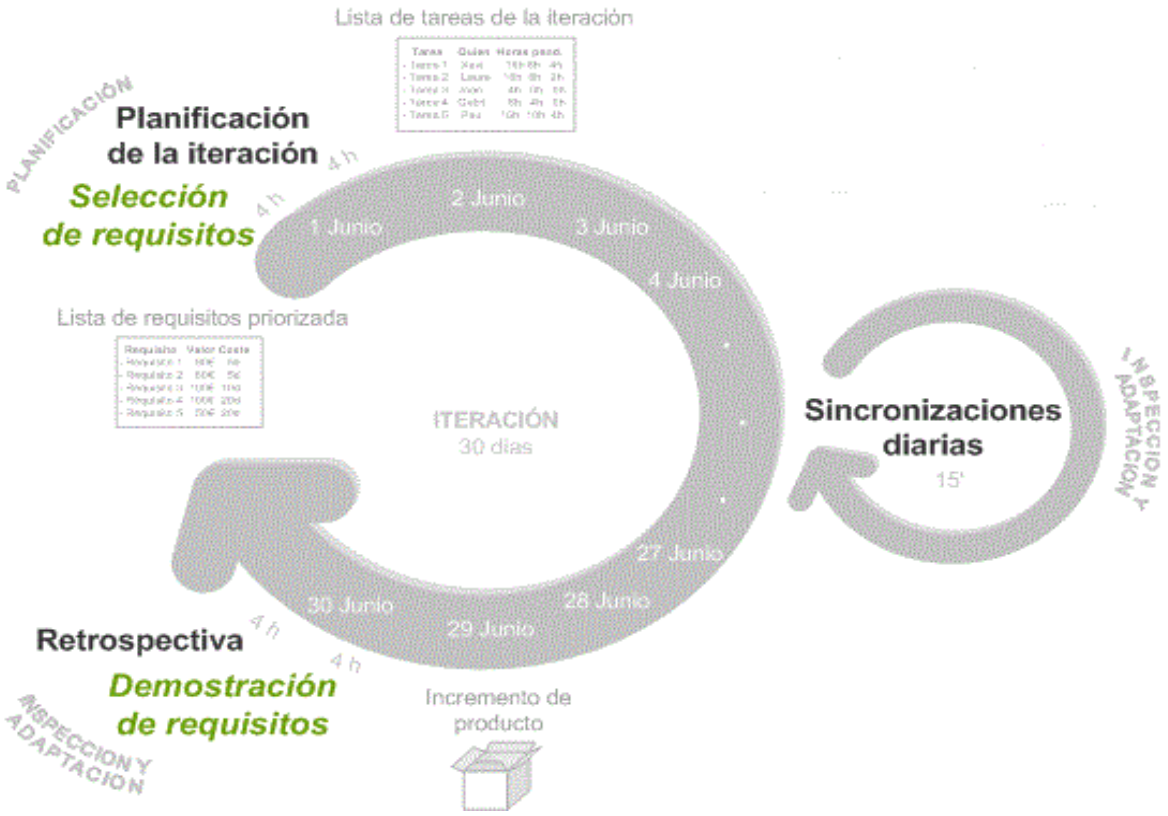

Figura 7. Proceso Scrum.

Fuente: Tomado de *Que es Scrum* [Imagen]. Proyectos Agiles. Recuperado de http://proyectosagiles.org

Actividades en el proceso Scrum:

- **Planificar la Iteración**

  Se hace la entrega de los objetivos del proyecto por parte del cliente al equipo de trabajo, allí se establecen los requisitos prioritarios para ser entregados al final de la iteración. Los miembros del grupo elaboran un plan de tareas de

manera conjunta y son asignadas internamente para el desarrollo de los requisitos a ser entregados en dicha iteración.

- **Ejecutar la iteración**

  Entre los miembros del equipo se realiza una inspección del desarrollo de las tareas con el objetivo de estimar el progreso en el desarrollo, por lo general se tienen en cuanta preguntas como ¿qué se ha hecho?, ¿qué se va a hacer? y ¿qué posibles impedimentos tiene o se tendrán? Además de las inspecciones internas del equipo, el líder del equipo (*Scrum Master*) también realiza su intervención en aras avanzar con los resultados sin disminuir la productividad.

- **Revisar y adaptar**

  El equipo junto con el cliente hacen la revisión exhaustiva de lo entregado en dicha iteración, si el producto es aprobado se pasa como incremento y se realizan las adaptaciones necesarias para re planificar el proyecto, es decir definir la siguiente iteración. El equipo de trabajo también analiza el desempeño durante este proceso de modo que se puedan resolver cualquier tipo de problema encontrado en busca de una mejora continua para posteriores iteraciones.

Técnicas usualmente aplicadas en Scrum

Historias de Usuario

Las user stories (Véase Figura 8) representan requerimientos desde el punto de vista del cliente donde se expresa el quien, el qué, el porqué de dicho requerimiento así como el nivel de satisfacción.

Figura 8. Características de una Historia de usuario.

Fuente: Tomado de *Que es Scrum* [Imagen]. Proyectos Agiles. Recuperado de http://proyectosagiles.org

En general esta técnica aclara la característica del requerimiento para evitar trabajar varias veces en la misma sección debido a malentendidos.

Tablero de tareas

Permite gestionar las tareas necesarias para llevar a cabo un objetivo dentro de una iteración y a través del proyecto. El equipo de desarrollo durante el desarrollo del Sprint selecciona la lista de tareas y las organiza en una tabla de modo que se pueda ver la evolución de cada una de ellas a medida que la iteración está en desarrollo como se observa en la Figura 9.

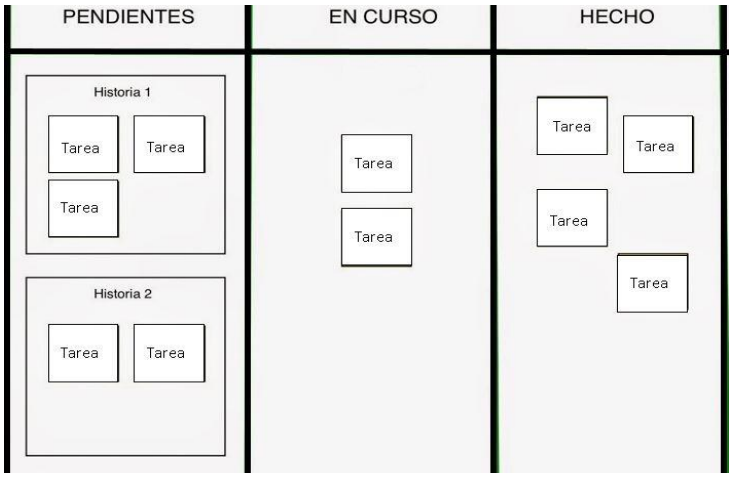

Figura 9. Tablero de tareas en Scrum.

Fuente: Elaboración propia

De esta forma el equipo tiene clara visibilidad del estado de cada una de las tareas en las sincronizaciones diarias correspondientes al Sprint actual.

Ventajas

- EL Product Owner o cliente puede hacer uso del producto desde su primera iteración, permitiéndole cambiar o plantear nuevos objetivos.
- Permite la elaboración de productos en diferentes tipos de proyectos de manera rápida, flexible y transparente, de modo que se acerque en casi su totalidad a lo deseado por el cliente.
- Los productos finalizados entregados al terminar cada sprint son de alta calidad debido a que el enfoque principal del equipo de desarrollo es el problema del usuario e ir trabajando de la mano de este a lo largo de todo el desarrollo y la prioridad está en aquellos módulos que agreguen mayor valor a dicha organización.
- Posibilita el trabajo en equipo manteniendo comunicación constante (cara a cara) con cada uno de sus miembros por medio de reuniones diarias, retroalimentación constante y transparencia con relación a lo que el equipo debe hacer cada día así como lo que lo que se entregara al cliente al final de cada sprint.
- Debido a que el proceso es iterativo y los lapsos de tiempo acotados el equipo de desarrollo puede enfocarse en funcionalidades especificas dentro de cada periodo.

Desventajas

- Si no se establece una fecha para la terminación del proyecto se pierde claridad de acuerdo al alcance y puede que el equipo continúe generando funcionalidades sin dar por terminada la entrega.
- Si los miembros que conforman el equipo no están completamente comprometidos o no se sienten a gusto puede que no se pueda finalizar dicho proyecto de forma exitosa.

- El hecho de que este marco de trabajo difiera de otros puede que impida su aplicación generando resistencia en determinadas personas.

2.2.1. Caso de éxito Spotify

Un ejemplo “tangible” acerca del éxito en la aplicación de métodos agiles para el desarrollo software móvil es el de la empresa Spotify. Consciente que compañías como Google o Apple deben su éxito al uso de metodologías para guiar sus procesos de desarrollo, decidieron adaptarse sistemáticamente a Scrum puesto que el principal interés de aplicar este tipo de metodología es que sabían que para no ser derrotados por corporaciones de esta categoría debían ser más rápidos, baratos y mejores. Gracias a Scrum caracterizaron la estructura ágil que les permitió implementar la aplicación del mismo nombre empleada para reproducir música vía Streaming (que se puede usar a la vez que se descarga) que goza de gran popularidad en la actualidad, en la Figura 10 se observa la aplicación disponible en Google Play y demás tiendas online.

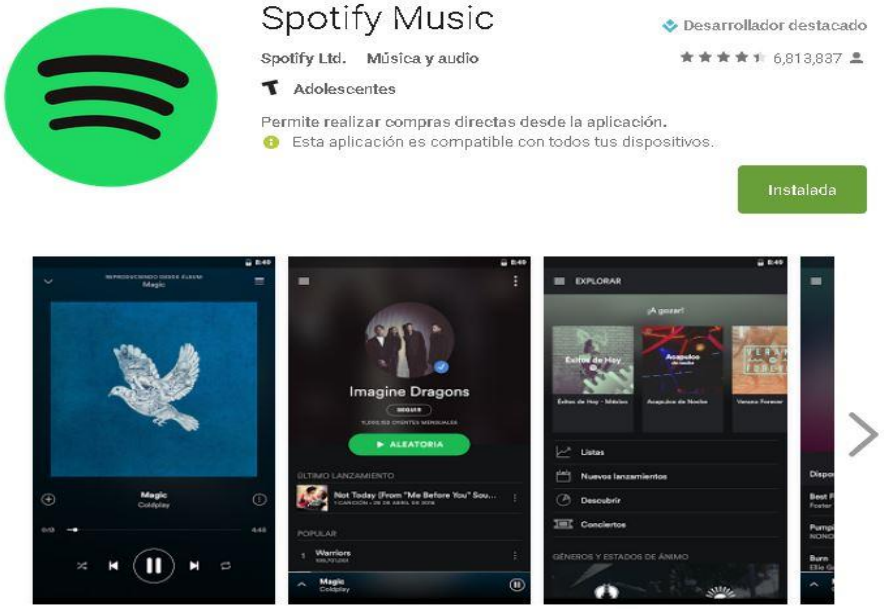

Figura 10. Aplicación Spotify.

Fuente: Google Play Store, Spotify. Recuperada de https://play.google.com/store

Se hizo hincapié en el rol de Scrum Master pero adaptado como Ágil Coach (algo así como un con consultor ágil del cambio), el Scrum Master se centra en su equipo único de trabajo y supervisa el desarrollo siguiendo la metodología mientras que el Agile Coach tiene una visión global de la empresa ayudando a en todos los departamentos y aportando a la superación profesional de los miembros del equipo. De éste modo el Agile Coach no define como resolver los problemas sino que identifica los posibles conflictos y notifica al equipo de modo que entre todos puedan decidir la mejor estrategia a seguir.

El enfoque de Spotify es más ágil que Scrum (Kniberg, 2014), es decir, le interesan los principios agiles mas no cerrarse a seguir los practicas específicas de Scrum; definen "Autonomous Squad", los cuales son pequeños equipos multidisciplinares autónomos y auto organizados de al menos ocho personas que se sientan juntas y tienen un inicio y un final para cada una de las cosas que construyen (diseño, despliegue, mantenimiento, operaciones, etc.), la autonomía del grupo permite decidir que desarrollar, como desarrollarlo y como trabajar juntos mientras lo hacen. Así que ¿Por qué darle autonomía al equipo?, bueno porque es motivador y personas motivadas construyen mejores cosas. Además la autonomía permite rapidez, ya que las decisiones se toman dentro del Squad y nos es necesario consultar con directores, gerente u otras dependencias. Todos estos squads cumplen sus propias funciones pero necesitan estar alineados con los demás squads, la estrategia de producto y prioridades de la compañía.

Spotify hace énfasis en las personas y mantenerlas motivadas, tienen una fuerte cultura de respeto mutuo, los desarrolladores hacen comentarios sobre el buen trabajo de sus compañeros y se guardan los créditos por el suyo, además si es necesario pedir ayuda se obtiene mucha de manera rápida, todos manejan la mentalidad de que están en el mismo barco y necesitan del trabajo de todos para obtener el éxito.

2.3. eXtreme Programming (XP)

La programación extrema es otro procedimiento de desarrollo ágil orientado a objetos, al igual que con los otros se basa en principios como la simplicidad, comunicación, retroalimentación y el valor. Promueve el trabajo en equipo haciendo de las relaciones interpersonales una de sus principales preocupaciones si se desea obtener el éxito de un proyecto. Una vez más establece la comunicación continua con el cliente y el equipo como unos de sus puntos fuertes la lograr fluidez en el entendimiento de los requisitos y tareas del proyecto, lo cual permite enfrentar con buena cara los cambios.

Al hacer uso de este tipo de procedimiento se tiene como objetivo grupos pequeños o medianos para la construcción del software donde los requisitos sean algo ambiguos o consideran un gran riesgo. Sugiere como lugar de desarrollo para el proyecto un espacio amplio pero no seccionado donde puedan estar tanto el equipo como el cliente de modo que sea mucho más sencilla la comunicación y halla una mejor dinámica entorno al proyecto a enfrentar (Beck, 2000).

Prácticas del procedimiento XP:

- **Diseño simple:** Se aplica buscando sacar el mayor valor de negocio con la producción de las características más importantes en el menor tiempo posible por medio de programación sencilla pero que cumpla a cabalidad los requerimientos establecidos.
- **Metáfora:** En los procedimientos agiles no es posible establecer una clara definición del sistema de manera temprana debido a que se asume que esta se va recreando de manera evolutiva, por ello tanto el cliente como el equipo manejan una metáfora general, la cual no es más que una historia de cómo se verá el sistema completo pero que brinda a todos una visión general del proyecto.

- **Entregas pequeñas:** La entrega de aproximaciones rápidas del sistema que sean funcionales significan un resultado de valor para el negocio.
- **Pruebas:** Se aplican pruebas unitarias y funcionales que son establecidas por el cliente previamente a la generación del código. Se aplican pruebas constantes como una forma de asegurar calidad en el producto, así como evitar posibles problemas más adelante.
- **Refactorización:** Esta actividad permite remover código duplicado, simplificarlo, y de a poco hacerlo más flexible con respecto a posibles cambios futuros. Debido a que no es posible crear código con precisión desde el inicio es necesario este tipo de reestructuración continua ya que así evoluciona conforme la funcionalidad del producto avanza.
- **Propiedad colectiva del código:** El código utilizado para el desarrollo del producto no pertenece a una sola persona, el código se elabora entre todo el equipo y le pertenece a todos, cualquier programador puede cambiar e interactuar con cualquier parte del código en cualquier momento. Este tipo de práctica permite motiva a que todos los programadores aporten sus ideas en cualquier segmento del desarrollo y así evita que miembros del equipo se hagan imprescindibles si es necesario abarcar cambios en código que haya sido manipulado por estos.
- **Programación en parejas:** Para esta práctica dos programadores deben trabajar en equipo haciendo uso del mismo equipo de cómputo, la idea es que también la conformación de las parejas cambie, es decir no siempre van a estar los mismos dos programadores. Esto permite que muchos errores sean detectados puesto que entre estos dos miembros del equipo van a estar constantemente haciendo auto crítica y revisión del código, los diseños son mejores, se comparten conocimientos, todos los desarrolladores están al tanto de la totalidad del sistema, hay menos tiempo muerto ya que si uno de

ellos se siente agobiado o cansado en algún punto le permite al otro trabajar esa instancia.

- **Integración constante:** Cada una de las nuevas partes del código son integradas y testeadas una vez que están listas, de esta forma el sistema puede ser implementado varias veces por día
- **Semana laboral de 40 Horas:** Buscando el bienestar de los programadores se establecen horarios no superiores a las 40 horas semanales y en el caso de ser necesarias horas extras no será posible hacerlo durante 2 semanas consecutivas. Esta norma permite que los desarrolladores trabajen frescos y con mayor comodidad permitiendo que produzcan código de mayor calidad.
- **Cliente en Sitio:** En los desarrollos agiles es necesaria la presencia constante del cliente, de modo que los desarrolladores puedan hacer cualquier tipo de pregunta, resolver inquietudes, establecer prioridades o verificar desarrollo de funcionalidades. Por lo general aunque se cuenta con la presencia de éste, es muy difícil que esté completamente de lleno en el desarrollo del proyecto.

La Figura 11 muestra el ciclo de vida Xp con sus etapas y actividades:

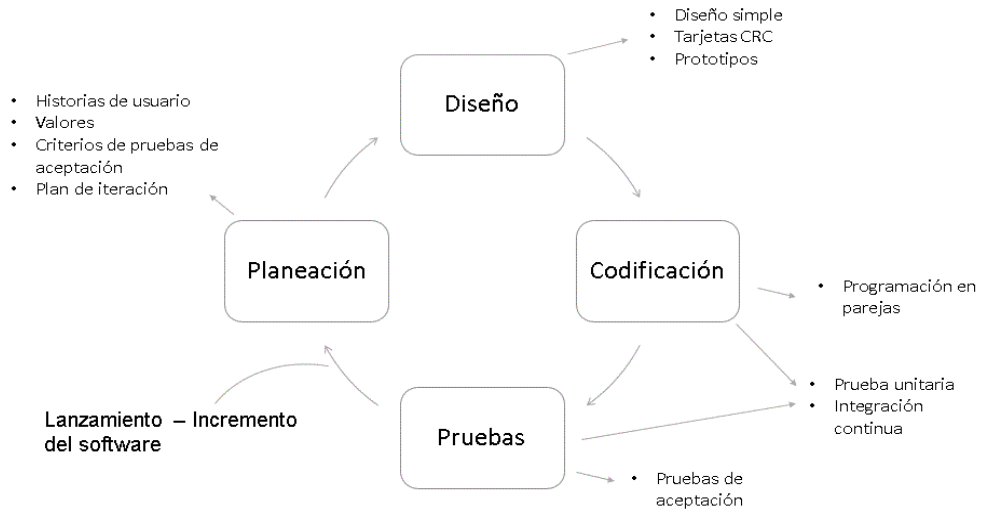

Figura 11. Ciclo de vida metodología eXtreme Programming.

Fuente: Elaboración propia.

Planeación: Una vez reunidos los involucrados en el proyecto se establecen por parte del cliente los requisitos del sistema, además los desarrolladores se familiarizan con las herramientas y las practicas que se utilizaran durante el proceso. Se establece una metáfora como modelo inicial del sistema. En este punto la recolección de esta información y el cambio de fase estarán determinado por el tiempo necesario para que los desarrolladores se familiaricen con la tecnología.

- Historias de usuario
- Iteraciones
- Entregas pequeñas
- Roles
- Ajustar XP

Diseño: Se establecen los requisitos más importantes y su alcance, se estima la duración necesaria para cada historia de usuario de modo que se pueda establecer un cronograma con dichas tareas. Como no es posible tener un diseño completo y sin errores desde el principio se establece una metáfora a partir de las historias para dar un panorama general del proyecto y que no sea muy extenso ya que está expuesto a cambios a lo largo del desarrollo. Se fija el número de iteraciones y depende del cliente decidir cuál historia se desarrollara en cada iteración. Se establecen las pruebas funcionales para cada iteración de modo que al finalizar la última iteración el sistema esté listo para producción.

- Simplicidad de diseño
- Metáfora del sistema
- Tarjetas CRC.
- Refactoring.

Desarrollo: Buscando obtener software de alta calidad se realizan pruebas extras y comprobaciones del funcionamiento del sistema antes de poder liberar al cliente. En este punto se revisa si necesitan incluirse nuevas características mientras esta en vigencia esta fase. Todas las propuestas, cambios o sugerencias son documentados para posteriores implementaciones. Se tiene especial cuidado con los aspectos referentes a la rotación de programadores o la programación en parejas.

- Cliente presente.
- Codificar pruebas.
- Integración.

Pruebas: Como con cualquier producto software su liberación dependerá de los resultados a las pruebas, y estos resultados a su vez dependen de la buena elaboración de las pruebas antes de ser aplicadas, si el proyecto paso por buenas prácticas de refactorización incrementa la obtención la posibilidad de pasar las pruebas con el 100 por ciento. En el caso de que el código no cumpla con las pruebas en su totalidad será necesario encontrar la falla del sistema e implementar una solución.

- Pruebas unitarias.
- Pruebas de aceptación.

Actualmente este tipo de metodología es aplicada por empresas de gran trayectoria como lo es Microsoft y Hewlett Packard en proyectos de desarrollo de software o en la gestión de proyectos internos.

Ventajas:

- Permite un estilo de programación muy organizado.

- Muestra eficiencia en los procesos de planificación y pruebas.
- Tiene antecedentes con bajas tasa de errores.
- Mantiene la satisfacción del programador.
- Fomenta una buena comunicación.
- Facilita los cambios.
- El cliente define y controla las prioridades.
- Obtiene buenos resultados gracias a la realización constante de pruebas.

Desventajas:

- Es muy recomendable para proyectos a corto plazo.
- Presenta inconvenientes de aceptación por parte de algunos programadores.

2.4. Test Driven Development (TDD)

El desarrollo orientado a pruebas (TDD) (Beck, 2002) guía a los desarrolladores a través del proceso y los enfoca en el producto final y las practicas necesarias para obtener una buena calidad. En éste las partes del código deben pasar diferentes pruebas previamente establecidas antes de pasar a producción, es ahí donde radica la importancia de este método, ya que cambia el razonamiento de los desarrolladores previniendo posibles errores, agilizando la obtención de resultados y aumentando la calidad del sistema propuesto. Su principal enfoque gira entorno a mejorar la productividad pero debido a sus aportes en la calidad de los productos se hizo llamativa su aplicación en conjunto con otras metodologías agiles como apoyo a diferentes tipos de proyectos. Su importancia para el desarrollo de software móvil radica en el refinamiento del código y la eliminación de errores, para esto el ciclo del desarrollo conducido por pruebas plantea las siguientes tareas posteriores a la captación de los requerimientos:

- **Seleccionar requisito:** Se escogen los requisitos esenciales y fácilmente implementables.

- **Escribir una prueba:** En este paso se debe entender por completo el requerimiento de modo que se pueda visualizar, desde el punto de vista del usuario, la totalidad de la funcionalidad para poder definir las posibles pruebas para tal requisito.
- **Verificar fallo en prueba:** En este paso si la tarea no falla es porque la prueba quedo mal implementada o el requerimiento ya existía.
- **Implementar:** Crear el código más simple que pueda hacer que la prueba funcione.
- **Refactorizar:** Se elimina cualquier duplicación de código, se avanza haciendo pequeños cambios hasta que funcione.
- **Actualizar requerimientos:** Se van descartando los requerimientos que ya pasaron las pruebas, si es necesario también se agregan nuevos a la lista.

En la Figura 12 podemos observar el proceso en las iteraciones de TDD.

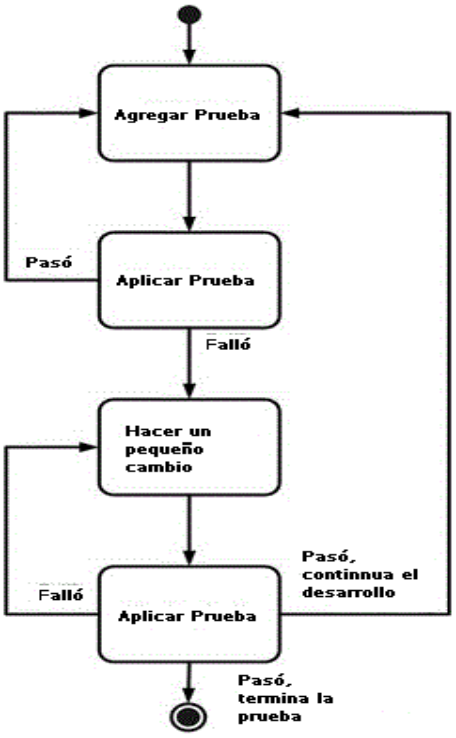

Figura 12. Representación de los pasos en TDD.

Fuente: *Introduction to Test Driven Development*. (s.f) Adaptado de http://agiledata.org/

2.5. Mobile-D

Metodología creada hace ya varios años como parte de un proyecto finlandés. En esencia es una mezcla de diferentes técnicas aplicadas para el desarrollo de software, sin embargo el uso en conjunto de diferentes modelos agiles logro una contribución original al escenario de las aplicaciones móviles, lo cual se puede detallar mejor en Mobile-D: an agile approach for mobile application development (Abrahamsson et al., 2005). Fue creada cuando apenas empezaba a crecer el entorno móvil, entonces el principal enfoque era la necesidad de lograr ciclos de desarrollo rápidos para equipos pequeños que no excedieran los 10 integrantes trabajando en un mismo espacio físico. Para esto, la propuesta se basa en soluciones agiles como eXtreme Programming (XP), Crystal Methodologies y Rational Unified Process (RUP), de los cuales hacen uso de las prácticas para el desarrollo, la escalabilidad de los métodos y la base del ciclo de vida respectivamente.

El proceso se divide cinco etapas o fases (Figura 13), a diferencia de la primera fase, las demás emplean tres días para preparar cada iteración, uno de planificación, otro de trabajo y el último para liberación.

- **Exploración:** En esta fase se generan los pasos a seguir dependiendo de las características del proyecto todo con la participación activa del cliente. En la formulación de este plan se recogen los requisitos primordiales para el desarrollo y se establecen los procesos asociados a cada uno de estos, de esta forma se obtendrá una visión general de lo que se espera obtener al finalizar así como su alcance y las funcionalidades que deberá incluir el producto.

Figura 13. Fases metodología Mobile-D

Fuente: Elaboración propia.

- **Inicialización:** El equipo de desarrollo identifica el tipo y la cantidad de recursos que serán necesarios para el desarrollo, además de adecuar el procedimiento a seguir en las posteriores fases así como la preparación del mismo equipo.
- **Producción:** En esta etapa se repite iterativamente la planificación, trabajo y liberación hasta conseguir que todas las funcionalidades sean implementadas por medio de un desarrollo guiado por pruebas en función de verificar el correcto funcionamiento.
- **Estabilización:** En este punto se procede a integrar cada uno de los productos o módulos desarrollados en las iteraciones de modo que se pueda completar la aplicación final. Se verifica la calidad del producto y se finalizan los documentos relacionados con el proyecto.
- **Pruebas:** Esta fase, como en cualquier método tradicional, tiene como objetivo pulir el producto terminado, pasando por diferentes tipos de pruebas que permitan obtener una versión final estable y funcional del sistema. En

este punto se revisan todos los requerimientos del cliente y se corrige o elimina cualquier defecto encontrado.

Adicionalmente, una ventaja importante de Mobile-D en cuanto a externalización de software es el hecho de que obtuvo una certificación nivel 2 en CMMI (métricas. de aseguramiento de calidad); y sus ciclos de desarrollo se han actualizado en la actualidad a partir de la experiencia en diferentes casos de uso (Abrahamsson et al., 2004).

Ventajas

- Permite, al igual que los procedimientos agiles más destacados, la posibilidad de realizar cambios en proyectos a bajos costos.
- La entrega de resultados rápidos.
- Cumple los alcances establecidos asegurando un producto adecuado en el momento indicado.

Desventajas

- No ofrece buenos resultados si es aplicado en desarrollos grandes o altamente segmentados.

- Existe una dependencia entre la obtención de buenos resultados y la buena comunicación del equipo de desarrollo.

2.6. ANTECEDENTES

A continuación se presentan algunas de las iniciativas que intentaron mejorar o proponer metodologías enfocadas al desarrollo móvil, así como las investigaciones encontradas cuya relevancia está en la aplicación de procedimientos agiles para la creación de aplicaciones en la plataforma Android. No se tuvieron en cuenta los trabajos desarrollados que se enfocaban a aspectos del desarrollo de software de bajo nivel (orientado específicamente al como programar una aplicación), se tiene

especial atención a aquellos orientados a metodologías. Se describen cada uno de los trabajos ordenados cronológicamente.

2.6.1. Modelo hibrido para desarrollo ágil

La propuesta de Rahimian y Rasmin (2008) representa una primera aproximación a la consolidación de una metodología propia de entornos móviles, la cual se apoya en la combinación del desarrollo adaptativo (ASD) y el diseño de nuevos productos (NPD), con la cual deja un poco de lado la gestión de proyectos y se enfoca en la rapidez de desarrollo en un mercado de gran volatilidad y dinamismo como lo es el de los dispositivos móviles. En ésta propuesta, tal como podemos ver en la Figura 14, a partir de un ciclo de vida tradicional se incorporan características de tipo ágil de modo que al iniciarse las iteraciones algunas de las fases se segmentan para mitigar diferentes riesgos, es decir que a lo largo del proceso se atienden diferentes apartados como el análisis, prototipado y pruebas según como avance el desarrollo.

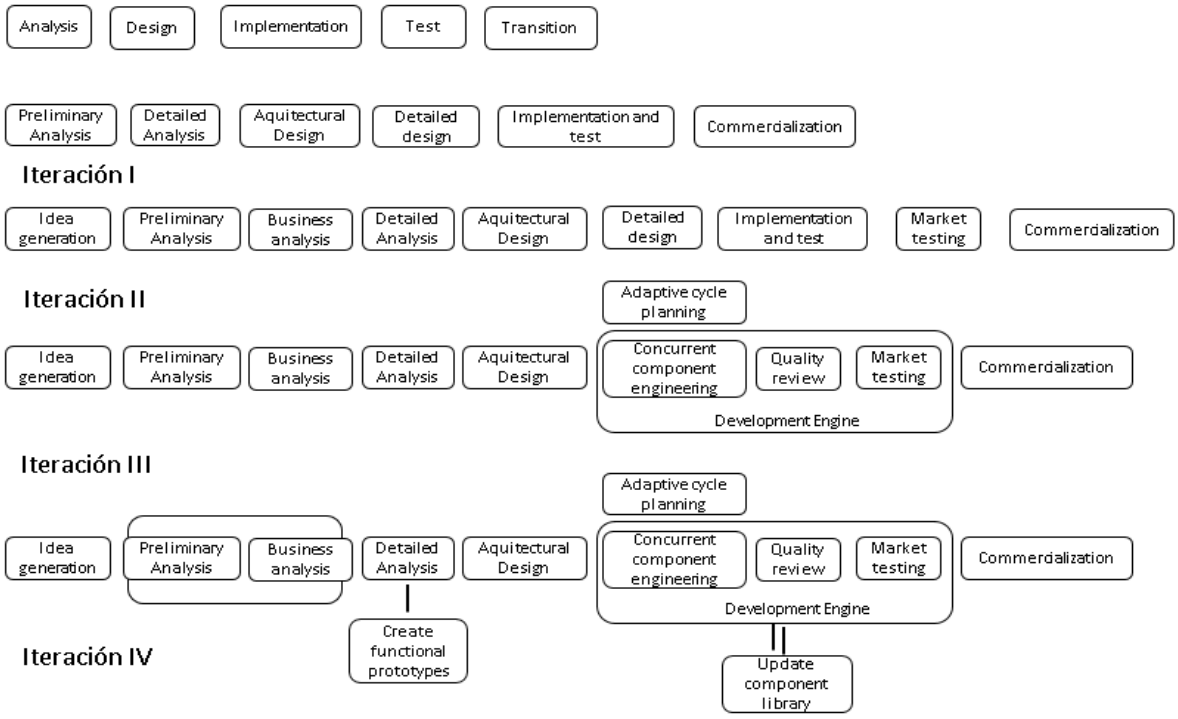

Figura 14. Fases e iteraciones modelo diseño hibrido

Fuente: Adaptado de Rahimian, V., Ramsin, R. (2008).

Uno de los atractivos del diseño hibrido es la inclusión de una fase de comercialización y que fusiona la implementación con las pruebas, lo cual está más orientado hacia escenarios del desarrollo de aplicaciones móviles.

2.6.2. Desarrollo de una aplicación para dispositivos móviles con sistema operativo Android aplicando Extreme Programming

Para el desarrollo de ésta tesis, tal como lo indica el título, se realizó un análisis sobre el proceso ágil extreme Programming para su posterior ejecución al desarrollo de una aplicación a la empresa Renase para el control de ventas y clientes. Con la culminación del proyecto se obtiene un alto grado de satisfacción por parte del autor, según afirma Mafla (2012):

> "*Se comprobó que la metodología de desarrollo ágil XP asegura la calidad del software durante todo su ciclo de vida y que sus valores son fortaleza de la misma. Durante el desarrollo de la aplicación se obtuvieron funcionales al término de cada iteración, lo cual representa el éxito en el desarrollo y el interés de la empresa Renase en continuarlo. El uso de sus valores como la comunicación, la simplicidad y la retroalimentación permitieron trabajar conjuntamente con la empresa ejecutando pruebas continuas de la aplicación con el usuario final, realizar un desarrollo más sencillo y flexible a cambios, del cual se obtuvo como resultado una aplicación óptima y conforme a las necesidades de Renase*". (p.82).

En la Figura 15 se pueden observar imágenes finales de la aplicación Renase propuesta.

Figura 15. Capturas de la aplicación Renase.

Fuente: Mafla, E. (2012). (p 98, 104, 108).

2.6.3. Análisis y estudio de las herramientas libres para el desarrollo de aplicaciones móviles en teléfonos celulares con sistema operativo Android, y la construcción de una aplicación para entretenimiento de usuarios en la carrera de informática y sistemas computacionales de la Universidad técnica de Cotopaxi

El principal objetivo de este proyecto era incentivar la incursión de los lectores y directamente implicados en el descubrimiento y aplicación de nuevas tecnologías, como las relacionadas con sistemas móviles, además de evaluar los conocimientos recientemente adquiridos en estas plataformas.

Se seleccionó la propuesta metodológica XP como esquema para implementar la aplicación denominada “Mida su inteligencia” (Maigualca, 2012), basada en preguntas y respuestas de razonamiento que finalizada la interacción representa la puntuación como un indicador de la inteligencia del usuario que la uso. Aunque este trabajo hace mayor referencia al análisis de las herramientas disponibles en el año 2012 para desarrollar aplicaciones móviles, también está documentada la ejecución de la metodología XP como guía para el desarrollo del proceso por lo cual se tuvo

en cuenta dentro de este estado del arte. El autor hace énfasis en características propias de la metodología por las cuales decidió abordarla para la puesta en marcha de este proyecto, características como el desarrollo con equipos pequeños sin necesidad de expertos, orientado más a la adaptabilidad que la previsibilidad, integrantes comprometidos y entrega de software que se necesita cuando se necesita.

Se rescata evidencia sobre la problemática que aun hoy en día afronta el proyecto actual, Maigualca (2012) concluye que:

> *Hasta la presente fecha no existe una metodología para el desarrollo de aplicaciones móviles, por lo cual se empleó la metodología XP que ayudo a llevar un proceso adecuado, destacando que no se empleó toda la metodología, sino que se acoplo según los requerimientos para el desarrollo*. (p.177)

2.6.4. Aprendiendo a desarrollar aplicaciones para Android con la metodología ágil Scrum: Un caso de estudio

En ésta investigación se realizó un ejercicio experimental sobre la creación de una aplicación móvil en Android en el área de matemáticas didácticas aplicando la metodología ágil Scrum por estudiantes del área de sistemas e informática (Roque, Negrete y Salinas 2013). En este caso estudio vemos que el análisis está orientado a la experiencia de los desarrolladores al aplicar la metodología Scrum, a lo largo del experimento se le presta especial atención al proceso de aprendizaje por parte de los programadores, la postura frente a la metodología, sus respuestas al poner en práctica las destrezas adquiridas profesionalmente.

Se resalta la información recolectada con las respuestas de los involucrados respecto a la situación real a la que se enfrentaron y como lo afrontaron, en especial lo referente a actividades como el trabajo y comunicación en equipo, la distribución de las tareas, la organización, toma de decisiones, etc. En general se observa un

buen nivel de satisfacción con la utilización de Scrum, más que nada en tanto a algunas de sus prácticas metodológicas, aunque sin embargo se notaron algunos inconvenientes según señalan Roque et al. (2013):

> *Los principales problemas o desventajas que los participantes encontraron en este desarrollo de software en particular fueron: diferentes niveles de habilidades y conocimientos, poca paciencia de algunos integrantes, la manera de socializar entre el equipo, dificultades en la toma de decisiones, inconvenientes con el entorno de desarrollo y con algunos aspectos avanzados de la sintaxis del lenguaje utilizado*. (p.9).

Este aspecto se debe tener en cuanta en cuanto a la conformación del equipo de desarrollo, en necesario que las personas involucradas estén preparadas para afrontar este tipo de proyectos y que cuenten con disponibilidad para trabajar en equipo. Los programadores deben tener las aptitudes necesarias para desempeñar las tareas que le sean asignadas, problemas como el que se presenta en este caso de estudio en que algunos miembros tenían problemas con la sintaxis de los lenguajes empleados puede ser crucial en cuanto a tiempos de desarrollo y la entrega oportuna de productos funcionales.

2.6.5. Metodología para el desarrollo de aplicaciones Móviles

Esta investigación tiene mayor relación con lo que pretende el presente proyecto, sim embargo, Gasca, Camargo y Medina (2013) proponen una metodología para el desarrollo de aplicaciones móviles de manera general, además fundamentan este método no solo en conceptos heredados del postulado sobre metodologías agiles, sino que también tienen en cuenta el enfoque de micro mundos interactivos de la ingeniería de software educativo con modelado orientado por objetos (ISE-OO) y la concepción de que las aplicaciones móviles deben garantizar el cumplimiento de las

necesidades de los usuarios teniendo criterios de calidad y éxito determinado por las 6M's.

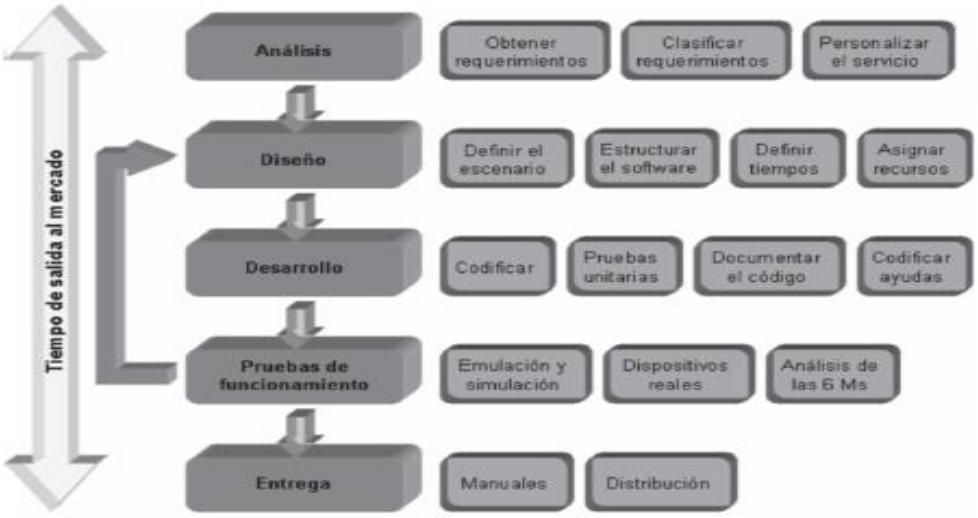

Figura 16. Etapas de la metodología DrMovil

Fuente: Tomado de Gasca, et al (2013), p. 24.

En la etapa de Análisis (Véase Figura 16), como en cualquier otro procedimiento, se obtienen, analizan y clasifican los requerimientos; en la etapa de Diseño plasman la idea de solución por medio de diagramas de Modelado de Lenguaje Unificado (UML), modelo vista controlador (MVC), diseño de capas, etc. En la etapa de Desarrollo se procede a codificar y realizar pruebas unitarias; se documenta el código según lo que se va implementando así como manuales de usuario, instalación y uso, A continuación en la etapa de Pruebas de Funcionamiento se hacen pruebas más rigurosas sobre funcionamiento de la aplicación, se emula el dispositivo móvil y se simula en diferentes tipos de escenarios, de ser necesario se debe regresar a la etapa previa para la corrección de errores. Además se incorporan pruebas en dispositivos reales de modo que se puedan medir factores como desempeño y rendimiento. Por ultimo en la etapa de Entrega se hace entrega del producto al cliente, junto con la documentación pertinente, se define la manera como se distribuirá el aplicativo.

Finalmente se concluye que la metodología propuesta permite afrontar de manera muy general proyectos para desarrollar aplicaciones móviles, sin embargo pese a que el resultado software cumplió su cometido, se evidencian muchas carencias en cuanto a la especificación de las tareas principales a tener en cuenta en desarrollos de esta índole, se obvian muchas características propias de los dispositivos para los cuales se desarrolla como es el caso del tipo de plataforma, los lenguajes que soporta, elementos de rendimiento y personalización por lo cual se llega a la obtención de fallas que pudieron haber sido evitadas desde el comienzo. Además, según evidencia Gasca, (2013): "*Los usuarios con sistema operativo Windows Mobile y otros no pudieron instalar, ni evaluar el servicio*" (p. 32), con lo cual queda claro que la mayor desventaja de esta metodología es que al ser de carácter universal sería necesario implementar una aplicación específica para cada plataforma de modo que se puedan prevenir errores de éste tipo y que según como está estructurada la metodología sería imposible hallar en la etapa de pruebas.

2.6.6. Aplicación móvil utilizando plataforma Android para mejorar la calidad del servicio de consulta de información de consumo eléctrico de la EEASA en la empresa Besixplus CIA LTDA

Este proyecto aborda nuevamente el uso de la metodología ágil Scrum como base para la implementación de una aplicación móvil sobre la plataforma Android, en este caso la problemática gira en torno a la ineficiencia para la consulta de información de consumo e historial eléctrico de la empresa Eléctrica Ambato Regional Centro Norte S.A (Tipantasig, 2013). Como la metodología propone, se delimito el alcance del proyecto calculando los Sprint necesarios para mantener un control sobre el proyecto.

En este caso, el autor describe el funcionamiento haciendo uso de diagramas de caso de uso para una mayor comprensión, y realiza la distribución de las tareas por medios de historias don describe las características de estas y la prioridad. El proyecto se hace algo largo y engorroso de leer debido a que cada una de las

iteraciones está documentada junto el código programado, sin embargo el autor se centra en las actividades descritas por Scrum para terminar la implementación.

Una vez más se evidencia algunas fallas en cuanto al entorno de desarrollo seleccionado para trabajar la aplicación según concluye Tipantasig 2013: "…*la utilización de la herramienta Eclipse fue de gran ayuda a la hora de desarrollar debido a que este programa es recomendado por Google para la creación de Apps, sin embargo al iniciar el proyecto de investigación no se contaba con los conocimientos suficientes*…" (p.132), es por este motivo que se hace necesario al análisis y selección de las herramientas desde el inicio del proyecto, usar herramientas conocidas por la persona que se va a encargar de la programación facilita en gran medida el desarrollo, además de que proporciona un entorno de comodidad.

2.6.7. Aplicación móvil para el seguimiento y control de las siembras de arrocera la esmeralda S.A – Alesa Móvil – Gaproa

Esta investigación tiene por objetivo la sistematización del proceso de control y seguimiento en la Arrocera La Esmeralda S.A (Vargas, 2014), gestionar la información principal de la empresa como visitas, insumos, posición geográfica, consulta de indicadores, estados y necesidades de siembra en tiempo real desde un dispositivo móvil gracias a una aplicación Android.

El desarrollo de la metodología se poco documentado a lo largo del texto, sin embargo se nota la organización en las etapas más generales, se muestra mayor atención por mostrar el código necesario para la implementación, sin embargo se definen por medio de diagramas las funcionalidades más representativas así como los almacenamientos de base de datos y esquema referencial de la aplicación. Además se resalta la aplicación de las historias de usuario para la recolección de requerimientos, esta característica de algunos métodos agiles es de gran importancia en cuanto a la definición de las tareas y el control de cada una de ellas, la Figura 17 muestra un ejemplo de las historias aplicadas en esta investigación.

Historia de usuario	
Número: 1	CONSULTAR CLIENTES
Modificación de Historia de Usuario : Ninguna	
Usuario: David Bastidas Vargas	**Sprint Asignada:** Sprint 1
Prioridad en Negocio: ALTA	**Tiempo Estimado:** 12 Horas
Riesgo en Desarrollo: MEDIO	**Tiempo Real:** 16 Horas
Descripción: El usuario podrá consultar clientes por los siguientes campos: Código o Nombre.	
Observaciones:	
Como Probarlo: Buscar 3 clientes por cada uno de los campos, Código y Nombre	

Figura 17. Historia de usuario usada en el desarrollo de Alesa Movil – Gaproa

Fuente: Adaptado de Vargas, D. 2014. (p. 30)

Una vez más se rescata una conclusión del autor de la investigación que tiene gran importancia dentro del contexto del proyecto actual, el cual afirma (Vargas, 2014):

> …*se deben tener en cuenta factores adicionales al momento de crear aplicaciones, como lo son los recursos limitados en algunos dispositivos, tamaños de pantallas, capacidad limitada de almacenamiento y sobre todo el soporte para las diferentes versiones del sistema operativo, en la actualidad existen patrones de diseño y herramientas que permiten asegurar todas estas características al momento de desarrollar*. (p.58)

2.6.8. Desarrollo ágil de una aplicación para dispositivos móviles. Caso de estudio: Taxímetro móvil

Este es uno de los casos de estudio más recientes y hace énfasis en la construcción de un prototipo de aplicación Android para monitorizar el recorrido al usuario de servicio público de taxi haciendo uso de la tecnología de GPS incorporada dentro del dispositivo móvil así como determinar una tarifa apropiada y el acceso a redes sociales desde la misma App en caso que quiera expresar inconformidad. En cuanto

al procedimiento a aplicar seleccionaron Scrum, así como modelos de UML para las etapas de análisis y diseño, además de técnicas estadísticas para evaluar requerimientos. En cuanto al porque se escoge esta metodología Babativa, Briceño, Nieto y Salazar (2015) afirman:

> …*se encuentran varias propuestas para el desarrollo de aplicaciones móviles, que van desde algunas modificaciones a las metodologías clásicas hasta la adopción de metodologías ágiles. En [3] se presenta el marco de trabajo denominado Mobile Application Development Life Cycle (MADLC, por sus siglas en inglés) que considera las etapas de identificación, diseño, implementación, prototipado, pruebas, despliegue y mantenimiento, pero en el que no se hace énfasis en el tratamiento ágil del desarrollo. En [8] se presenta una propuesta para la construcción rápida de aplicaciones móviles mediante un generador de código que facilita la implementación de tareas repetitivas (ejecución de ciertos comandos con el teclado, manejo de menús, conexión a una base de datos, envío de SMS), pero no se abordan los aspectos metodológicos. En [4] se propone un framework genérico para desarrollar aplicaciones móviles sobre diferentes sistemas operativos, pero tampoco discute el tema metodológico*. (p.2)

Según los resultados obtenidos se establece un gran nivel de confianza en la aplicación de la metodología Scrum, la cual permitió gestionar el proceso por medio de Sprint e historias de usuario iterativas e incrementales que facilitaban los cambios sobre funcionalidades así como modificaciones o creación de nuevas e incluso la eliminación de éstas. También muestra la aplicación de estrategias adicionales para validar y verificar tareas como revisiones continuas, pruebas de unidad y sistema, estadística, entre otras. Para mayor información revisar la documentación completa del proyecto en.

2.6.9. Desarrollo de una aplicación móvil que permite a los docentes y estudiantes de la Universidad Central del Ecuador acceder a las bases de datos científicas

Esta investigación tiene objetivo principal el desarrollo de una aplicación que permite consultar de manera ágil las bibliotecas virtuales que proporciona la Universidad Central del Ecuador, se desarrolló para las dos plataformas más populares en la actualidad como lo es Android e iOS. En cuanto al procedimiento para seguir el proyecto se analizaron las metodologías Scrum, XP y Mobile-D, seleccionando finalmente ésta última. Se destaca la selección de herramientas como Sesha Touch y Phonegap (Tumipamba, 2016) para desarrollar la aplicación ya que estas permiten el trabajo con varios sistemas operativos (multiplataforma).

A lo largo del documento se evidencia la aplicación del procedimiento Mobile-D que al igual que en otros casos se adapta según el tipo de proyecto y se dejan de lado algunas de las actividades descritas en el modelo. Para mayor información pueden revisar el documento en.

2.6.10. Guía metodológica ágil, para el desarrollo de aplicaciones móviles “AEGIS-MD”

La investigación tiene como objetivo proponer una guía metodológica que permita organizar las medidas necesarias para el desarrollo de aplicaciones en el entorno móvil teniendo como referencia los métodos XP, Scrum y TDD (Amaya, 2015).
La Figura 18 expone las etapas descritas por la propuesta junto con las actividades definidas por el autor para el cumplimiento de cada fase.

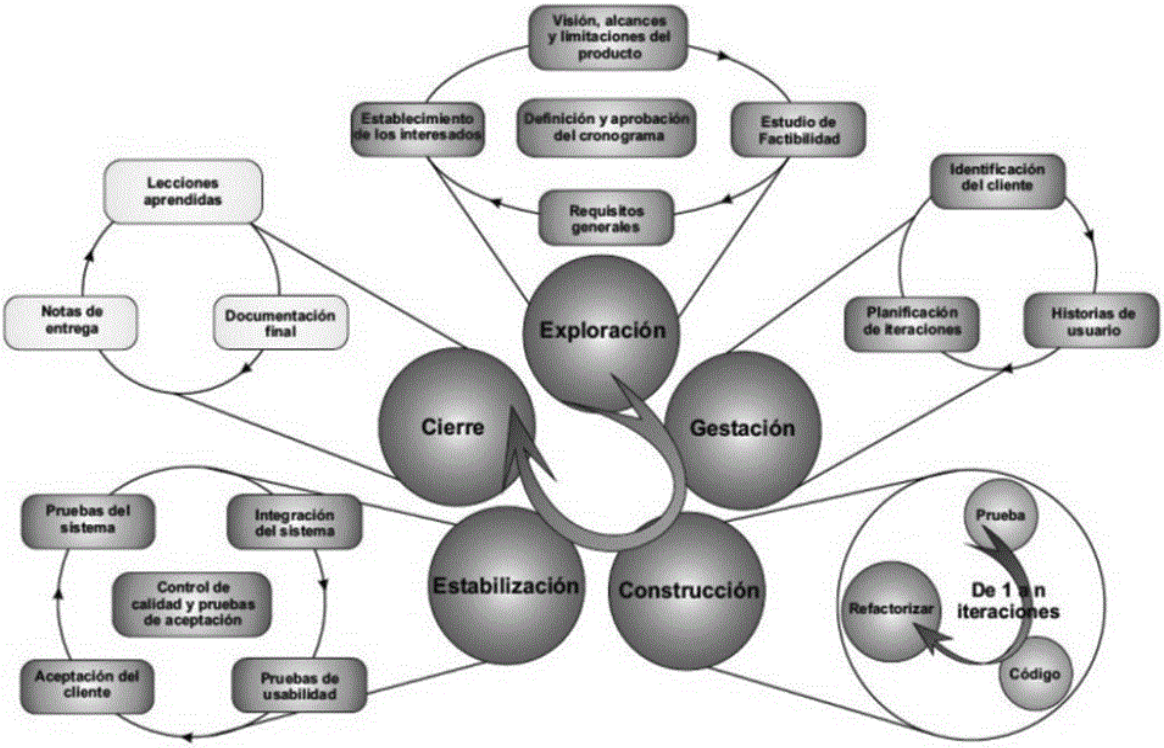

Figura 18. Guía metodológica AEGIS-MD

Fuente: Tomado de Amaya, Y. (2015). Guía metodológica ágil, para el desarrollo de aplicaciones móviles "AEGIS-MD" [Figura]. *Revista de investigaciones UNAD, 14, (1)* (p.107)

El enfoque de la propuesta denominada AEGIS-MD busca mejorar la recolección de los requerimientos cambiantes y la gestión de riesgos, desglosa el proyecto en iteraciones pequeñas para desarrollarlas bloque a bloque así como la reducción de tiempo de desarrollo por medio de la corrección temprana de errores.

Al igual que las demás metodologías agiles, AEGIS-MD se basa en principios de colaboración y trabajo en equipo, definición de pruebas tempranas y en mantener el buen ánimo entre los integrantes a lo largo del proceso.

2.6.11. Modelo para el desarrollo de aplicaciones nativas en Android basado en mejores prácticas, metodologías agiles y elementos del área interacción humano-computadora

Esta investigación enfoca diferentes prácticas para definir finalmente una propuesta de modelo que permita el desarrollo de aplicaciones nativas Android, en general abarca tanto características de los modelos agiles como línea principal para administrar el proyecto y la aplicación de mejores prácticas en busca de mantener criterios en calidad, eficiencia y escalabilidad del software. El modelo propuesto (Veloz, 2016) describe cinco etapas tal como se observa en la Figura 19 y en cada una de ellas se reflejan las tareas necesarias para continuar con el proceso.

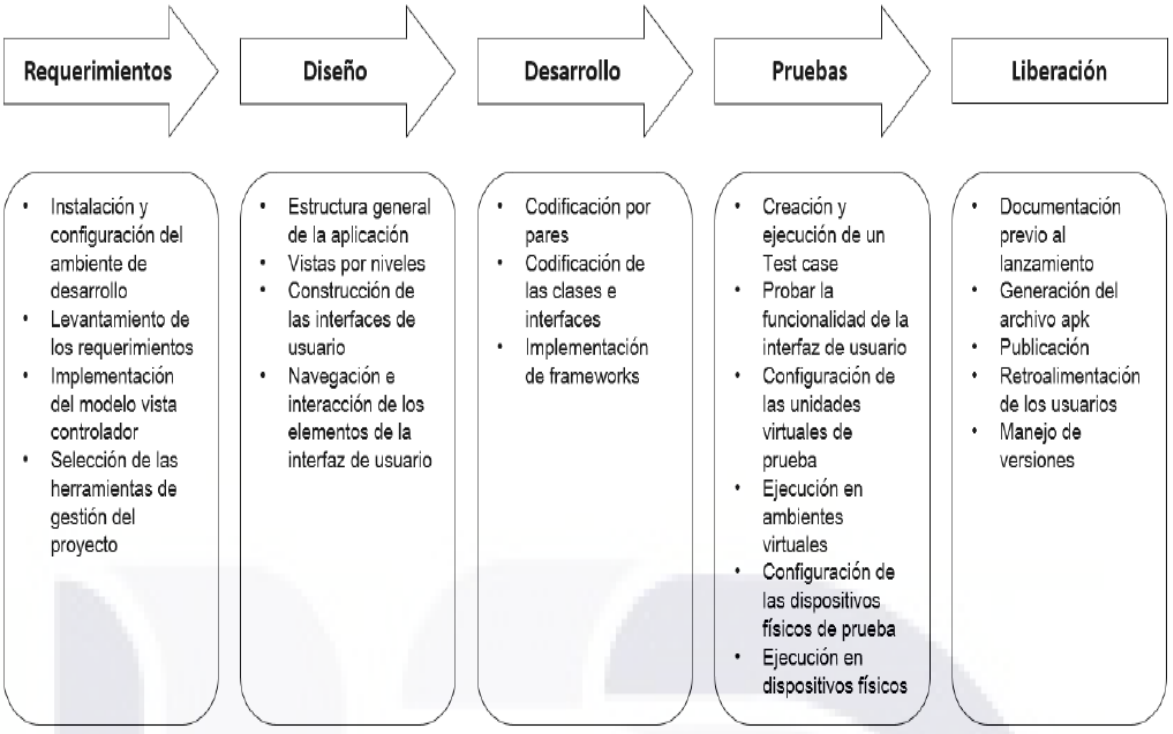

Figura 19. Etapas del modelo propuesto en la investigación. (Véase 2.2.11).

Fuente: Tomado de Veloz, C. (2016) p. 64.

Sin embargo, aunque el proyecto parece abarcar todo el contexto del presente, centra su atención en las aplicaciones de tipo e-health (eSalud- prácticas de la salud apoyadas en tecnologías de la información y las comunicaciones TIC), por lo cual la consolidación del modelo lleva características ajustables a implementaciones de aplicaciones de este tipo, es decir, que la practica en general no maneja una generalidad en cuanto al desarrollo abierto en Android sino que caracteriza más las

aplicaciones para cuidados de la salud. A partir de esta investigación se recoge buena fundamentación sobre algunos aspectos que no se tenían en cuenta en un principio pero que al final permiten que el desarrollo de este proyecto sea más sustancioso.

La investigación aborda buenas prácticas en la propuesta, cada una de ellas descrita ampliamente en Veloz (2016), además se recoge buena fundamentación con respecto a la aplicación de una prueba a estudiantes concernientes al área de software los cuales analizaron y evaluaron la propuesta como una buena práctica para ser llevada a cabo en escenarios reales. En cuanto a los objetivos planteados se evidencia un resultado satisfactorio, además en cuanto al presente proyecto aporta información relevante que podrá ser comparada y estudiada en busca de áreas de oportunidad.

2.7. Análisis comparativo entre los procedimientos y metodologías descritos

Cuando se hablaba de procedimientos agiles algunas personas asociaban esto a la falta de documentación inclusive a falta de control y la verdad es que no es así, simplemente estas prácticas tiene otro tipo de prioridad, se enfocan en tareas que aporten valor al negocio y en las relaciones que permitan logran mayor calidad minimizando el impacto de aquellas que son menos imprescindibles, buscando así, conseguir el objetivo del proyecto y la satisfacción del cliente. Procedimientos agiles hay varios, existen otros que han sido creados a partir de la mezcla de las prácticas de los existentes, a estos se les denomina métodos híbridos o mixtos. Como saber ¿cuál método debo aplicar?, la verdad no es sencillo escoger uno como el más relevante, todo depende del tipo de proyecto que se va a desarrollar. Actualmente muchas empresas hacen uso de Scrum en sus proyectos y usan Kanban para la gestión, la mezcla de estas técnicas ha sido denominada por algunos como *Scrumban*, en otras compañías prefieren aplicar XP ya que con este logran mucha

más organización y está más orientado a la ingeniería y en otros proyectos mucho más grandes logran aplicar no solo Scrum sino que también usan XP para evitar cualquier tipo de desperfecto, lo cual es una muy buena idea ya que se complementan mutuamente estos procedimientos. Sin importar cuál sea usado todos se rigen por la filosofía del Manifiesto ágil, con lo cual a la larga siempre estarán cubiertos por sus postulados. En el caso del desarrollo móvil observamos que las practicas están enfocadas en estos procedimientos aunque no se suelen aplicar todos los métodos conocidos para estos tipos de proyectos, pese a que todos trabajan un pensamiento ágil no todos aportan el mismo valor ni permiten obtener los resultados esperados. En la Tabla 4 se comparan las características pertinentes en desarrollos agiles de las metodologías y procedimientos analizados anteriormente.

Tabla 4.

Matriz comparativa de las metodologías analizadas.

Característica	**Scrum**	**XP**	**Mobile-D**	**TDD**	**DrMovil**	**Modelo propuesto (Véase 2.2.11)**	**AEGIS-MD**
Heredan modelos			X		X	X	X
Documentación estricta					X	X	X
Proceso sistemático		X		X	X	X	X
Enfocado en los procesos					X		X
Enfocado en las personas	X	X	X				X
Resultados rápidos	X	X	X		X	X	X
Cliente activo	X	X	X			X	
Manejo de tiempo	X	X	X	X	X		X
Refactorización del código		X		X			X

Iterativo	X	X	X	X	X	X	X
Respuesta al cambio	X	X	X	X	X	X	

Nota: Elaboración propia.

De acuerdo a la información representada en la Tabla 4 observamos que los métodos analizados cumplen con características del enfoque ágil aunque no las cumplan todas, esto es debido a que cada método fue diseñado originalmente según una serie de parámetros que buscaban dar solución a diferentes problemas en determinada época. Además observamos que los métodos propuestos en algunas investigaciones y que tienen como base metodologías como Scrum y XP no necesariamente cumplen algunas características que éstos sí. Probablemente sea porque al buscar dar un enfoque hacia el desarrollo móvil algunas de las practicas o actividades propias de las metodologías debían ser adaptadas u descartadas. Sin embargo en la Figura 20 observamos cuales de los métodos analizados destacan en las características a tener en cuenta para elaborar la propuesta en este proyecto.

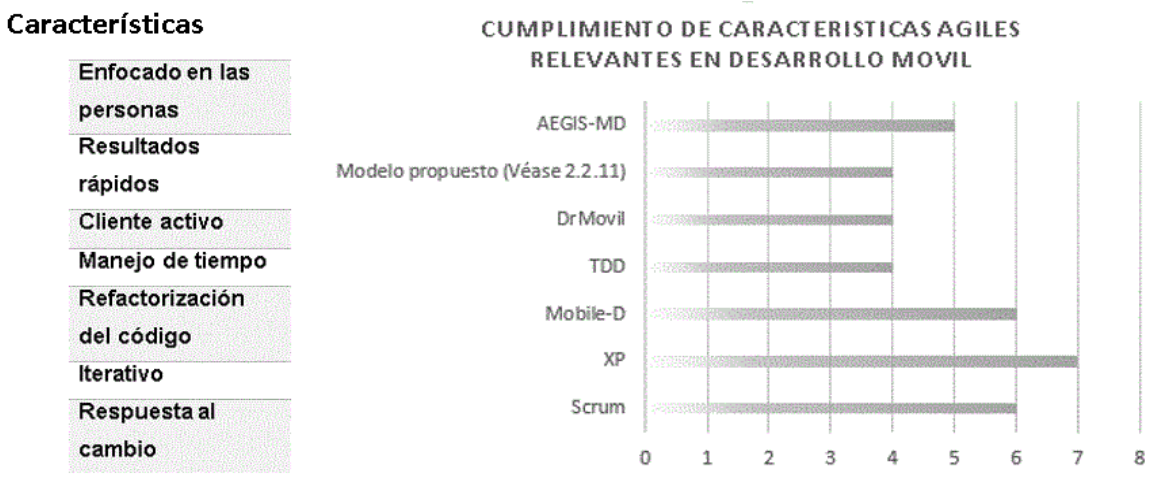

Figura 20. Características relevantes al desarrollo móvil.

Fuente: Elaboración propia.

Toda la información recopilada en la investigación permite, a grandes rasgos, satisfacer dos tareas de vital importancia para el cumplimiento de los objetivos, la primera identificar cuales procedimientos son aplicados en la actualidad para controlar el desarrollo de aplicaciones en dispositivos móviles con la plataforma Android y la segunda estudiar los ciclos de vida o etapas de estos procedimientos de modo que se puedan seleccionar las actividades y tareas que aportan valor al enfrentar las características propias de este tipo de software. Para esto último la Tabla 5 muestra en paralelo los procedimientos analizados y las características de sus etapas. Se definen los siguientes parámetros: 1. Proporciona soporte a la gestión del proyecto, 2. Describe lo que se debe hacer en cada etapa, 3. Define las actividades y tareas de la etapa.

Tabla 5.

Paralelo entre procedimientos y sus etapas.

	Especificación de requisitos			Diseño			Codificación			Test Unitarios			Test de integración			Test de sistema			Test de aceptación		
	1	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
SCRUM	X	X	X	X			X	X		X	X		X	X	X						
XP		X	X		X	X		X	X		X	X		X	X		X	X		X	X
MOBILE-D	X	X		X	X		X	X		X			X								
TDD					X	X		X	X		X	X		X	X		X	X		X	X
DRMOVIL	X	X		X	X	X	X		X	X											
AEGIS-MD	X	X	X				X	X					X			X	X		X	X	
MODELO E-HEALTH	X	X		X	X		X	X	X	X			X			X	X				

Nota: Elaboración propia.

De este análisis podemos ver que a pesar de ser metodologías y procedimientos basados en conceptos agiles, no todos se concentran de igual manera en cada una de las etapas del proceso. Por ejemplo, Xp es uno de los más sistemáticos, está orientado al seguimiento de las prácticas a lo largo del proceso por lo cual tiene presencia en todas las etapas mientras que TDD está más enfocado en la aplicación de pruebas. Sin embargo cada metodología permite la obtención de buenos resultados debido a que se puede decir que son ajustables, es decir, no necesariamente deben aplicarse al pie de la letra, se adaptan según el problema que deban enfrentar, además de que se pueden complementar entre ellas.

El entorno móvil hoy en día pasa por una etapa de gran apogeo, muchas empresas y personas en general dirigen su atención a este basto mercado buscando una forma de incursionar en el. La mejor opción es la elaboración de aplicaciones que aparte de poder compartirlas con el mundo entero también puede generar algunos ingresos extra, todo esto dependiendo del valor innovador que esta tenga además de la forma en que lo presente. Si se enfocan los esfuerzos en este aspecto entonces podemos decir que todo dependerá de la **idea** que genere o promueva embarcarse en un proyecto de este tipo. Una vez que la idea se tiene se encuentra entonces el problema que impulsa este proyecto, si se quiere desarrollar esta idea necesita de un procedimiento que le permita tener resultados de forma **rápida, sencilla y con eficiencia**. Pero, ¿Por qué?, porque no se es el único que puede estar trabajando en esta idea, sino se realiza de forma rápida alguien más lo hará.

*"**Sencillez,***
Que no ofrece dificultad, que carece de ostentación y adornos"

*"**Eficiencia,***
Capacidad de disponer de alguien o algo para conseguir un efecto determinado con el mínimo de recursos o en el menor tiempo"

Capitulo III: Procedimiento EfiSenDroid

A continuación se describe un procedimiento para el desarrollo de aplicaciones móviles concretamente para el sistema operativo Android (Véase Figura 21). Este procedimiento está basado en su totalidad en prácticas de tipo ágil y usa algunas de las características descritas por los procedimientos mencionados en el capítulo. Sus etapas se describen en una serie de pasos organizados que permiten tomar los aspectos más relevantes a tener en cuenta cuando se desea desarrollar proyectos cortos de este tipo.

Figura 21. Logo procedimiento EfiSenDroid (Opcional)

Fuente. Elaboración propia.

EfiSenDroid es un procedimiento ágil básico para el desarrollo de aplicaciones para *Android*. Su nombre hace referencia a dos criterios de vital importancia dentro del amplio mundo de los productos software para dispositivos móviles, en este caso se hace referencia a la **eficiencia** (a través del procedimiento lograr el desarrollo de una aplicación funcional con bajos recursos) y la **sencillez** (pocos pasos y proceso intuitivo).

Primeros pasos y conocimientos previos

Este procedimiento junto con la Internet a nuestro alcance permite proponer proyectos a corto plazo sin ningún tipo de obstaculización. La mayoría de las tareas aquí descritas proponen herramientas de fácil acceso y de usabilidad muy intuitiva para su desarrollo.

Gracias a que el proyecto concentra su atención específicamente en el sistema Android y no en entornos móviles en general, es mucho más sencilla la búsqueda y obtención de documentación que permita eliminar cualquier duda que se pueda presentar con respecto a la fase de Desarrollo. Sería recomendable tener conocimiento previo en cuanto al manejo de lenguajes de programación o en cuanto a la ingeniería software ya que eso facilitaría en gran medida la etapa de desarrollo además de evitar posibles demoras en cuanto al alcance del proyecto. Aun así, es posible desarrollar un producto de este tipo al menos con conocimientos básicos en programación Java, simplemente es cuestión de hacer uso de las herramientas de búsqueda en internet para aclarar dudas y alargar las fechas de entrega y finalización cuando se esté iniciando el proyecto (para usuarios inexpertos).

Sin embargo, se propone una lista opcional de requerimientos previos para poder aplicar este tipo de procedimiento:

- Poseer lógica como programador
- Tener conocimientos sobre Ingeniería de Software.
- Conocer y manejar lenguajes de programación como Java, XML, html5, CSS, JavaScript, entre otros.

Roles en EfiSenDroid:

Cliente: Es la persona que da origen a la problemática. A partir de él se extrae la información necesaria para lograr dar inicio al desarrollo, los requerimientos que se deben cumplir, modificar, agregar o eliminar así como la forma en que se deben comportar. Hace parte del equipo de trabajo, presente para afrontar las pruebas de cada artefacto a lo largo del proyecto.

Equipo de desarrollo: Conformado por personas con conocimientos en desarrollo software, si el proyecto amerita un trabajo multidisciplinar contar hasta con ocho

integrantes pero preferiblemente menos por cuestiones de organización y asignación de deberes, sin embargo este proyecto da evidencia de que el desarrollo puede ser afrontado hasta por una sola persona. La selección del equipo y su organización es primordial para la obtención de buenos resultados, se recomienda mantener prácticas motivacionales, resaltar el respeto mutuo y el buen trabajo en equipo.

En la Figura 22 se observan las etapas definidas por el procedimiento EfiSenDroid y las tareas relacionadas con cada una de éstas.

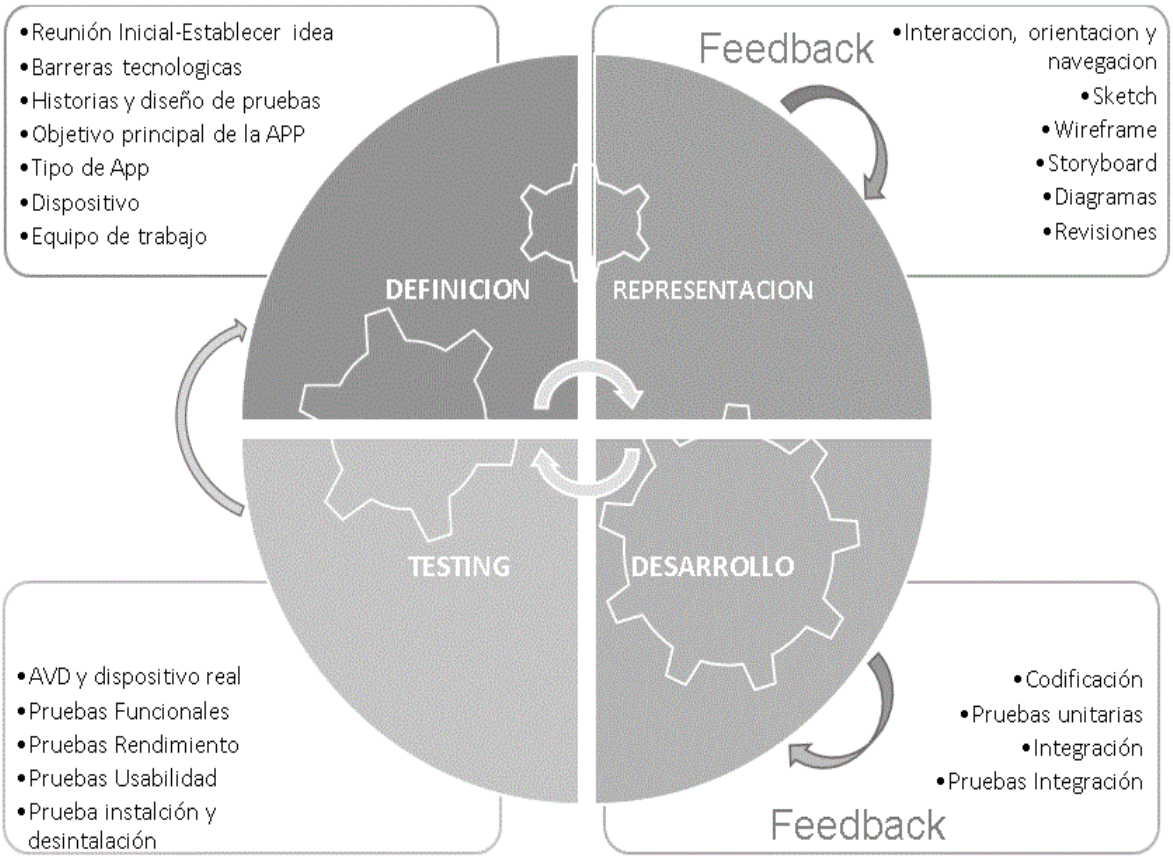

Figura 22. Procedimiento EfiSenDroid.

Fuente: Elaboración propia.

3.1. DEFINICION

En esta primera etapa se propone y aclara lo máximo posible la idea. Una vez que la idea es dada a conocer es necesario que el cliente empiece a desglosarla en pequeñas partes generando así los requerimientos que se convertirán en las tareas del equipo y los cuales darán la primera impresión real de lo que va a ser el proyecto.

*"Todo comienza con una **idea".***

3.1.1. Reunión y definición de la idea

Esta actividad se debe desarrollar junto con todos y cada uno de los miembros del equipo en una reunión formal. Se establece registro de cada uno de los requisitos dados por el cliente para más adelante generar una visión de lo que se debe hacer. Opcionalmente se puede dar un nombre para la aplicación y de ser necesario establecer como requisito un **Icono**.

3.1.1.1. Barreras tecnológicas

Es necesario que la idea presentada por el usuario sea analizada desde el punto de vista de la factibilidad y la viabilidad. Es de vital importancia que los desarrolladores expongan todas las dudas que surjan frente a lo expuesto por el cliente y que le hagan saber los posibles impedimentos que acarrearía su desarrollo. Se recomienda hacer uso de ejemplos en diferentes escenarios para aclarar en mayor medida los requerimientos de la propuesta, tanto por parte del cliente como del equipo de desarrollo. De este modo se cubren aspectos del ambiente externo que posiblemente no se hubieran tenido en cuenta antes del abordar el proyecto.

3.1.1.2. Historias: Cada historia de usuario representa un requisito del sistema, el cliente las expresa en lenguaje común y no son largas. Se deben hacer

las preguntas necesarias para aclarar cada una de las historias. En la Figura 23 se muestra la plantilla a utilizar para el desarrollo del proyecto.

Historia de Usuario EfiSenDroid	
HISTORIA N°:	**FECHA:**
USUARIO:	**ELABORADA POR:**
NOMBREDE HISTORIA:	
MODIFICACION A HISTORIA N°:	**VERSION:**
PRIORIDAD:	**RIESGO:**
DESCRIPCIÓN:	
PRUEBAS:	

Figura 23. Plantilla para Historias de usuario.
Fuente: Elaboración propia.

Historia N°: Identificador de las historias dentro del proceso, se asignan números únicos e irrepetibles de manera incremental para mantener la organización.

Fecha: Define la fecha exacta en que fue elaborada la historia. Por medio de ésta se establece un control cronológico.

Usuario: Es la persona que relata la historia, en la mayoría de los casos es el o los clientes del proyecto. En pocas palabras el usuario final que hará uso de la funcionalidad que se está definiendo.

Elaborada por: Persona encargada de redactar las características de la historia narrada por el usuario, en general hace referencia a los miembros de equipo de desarrollo.

Nombre de la Historia: Describe en forma de titulo la idea principal de la historia, por ejemplo: Registrar Usuario.

Modificación de Historia N°: Representa el número de la historia, previamente creada, a la cual se le hará alguna modificación.

Versión: Define la cantidad de veces que una misma historia ha sido tratada. La versión cero (0) hace referencia a la Historia original, en adelante representan modificaciones.

Prioridad: Asigna un valor (Alto-Medio-Bajo) a la historia de acuerdo a qué tan esencial es para el cliente, además permite al equipo de desarrollo definir la planificación mínima para el proceso. La priorización de las tareas ordena las historias de acuerdo al valor, costo y riesgo del requerimiento.

Riesgo: Asigna un valor (Alto-Medio-Bajo) a la historia de acuerdo a los posibles inconvenientes que podría representar la implementación de ésta dentro del desarrollo del proyecto.

Descripción: Explica en pocas palabras lo que se quiere o pretende con la implementación de la historia. Se recomienda responder "*como (rol) quiero (algo) para poder (beneficio)*".

Pruebas: Se establecen las pruebas que corroboraran la aceptación del artefacto desarrollado. Se describen de manera sencilla de modo que sea fácil aplicarlas para la revisión de las funcionalidades.

Permiten administrar de forma rápida esta información valiosa sin necesidad de tanta documentación formal además de dividir en pequeñas entregas. Suelen estimarse entre 10 horas y 2 semanas, relativamente hablando, para el desarrollo de cada historia, debido a que los modelos agiles y el desarrollo dirigido a dispositivos móviles no permiten una planificación fija por la incertidumbre que existe con respecto al cambio, por tal motivo es recomendable no establecer fechas límites estrictas para el desarrollo de las actividades.

3.1.1.3. Establecer Objetivo principal de la aplicación: Se especifica el objetivo central del proyecto, es decir, se describe con palabras ¿Qué va a hacer la app?, ¿Qué necesidad va a suplir?

3.1.1.4. Tipo de Aplicación y dispositivo: En este apartado se aclara el tipo de aplicación que se va a desarrollar, actualmente existen 3 tipos de aplicaciones.

- **App Nativas:** Son aquellas que son desarrolladas específicamente para un sistema operativo, se encuentran en su tienda de aplicaciones y deben ser instaladas directamente en el celular. Estas tienen acceso total en tanto a hardware y software, permiten mejor experiencia, pueden hacer uso de las características del celular (cámara, reproductor, galería, etc.), funcionan aunque no se disponga de conexión a internet o datos (offline).
- **App Webs:** Son mucho más sencillas de desarrollar y con los lenguajes tradicionales, son desarrolladas como páginas web normales pero optimizadas para responder a otros tamaños de pantalla. Son independientes del sistema operativo y funcionan a través del navegador siguiendo la ruta de una URL. No necesitan ser instaladas.
- **App Hibridas:** Como su nombre lo indica, estas combinan características de las dos anteriores. Pueden ser desarrolladas con lenguajes de las Webapp (HTML, Js, y CSS) lo que posibilita la multiplataforma y además también pueden hacer uso del hardware del celular.

Dispositivo al que va dirigida

En este apartado se especifica si la aplicación que se va a desarrollar será para Smartphone, Tablet, Android Wear, Android TV, Android Auto, etc. Actualmente los entornos de desarrollo permiten la definición del tipo de dispositivo y el sistema mínimo sobre el cual funcionaran por medio

del AVD (Dispositivo Virtual Android), de esta forma mientras se codifican las características se pueden hacer pruebas donde se emulara virtualmente las características como el tamaño de pantalla del dispositivo seleccionado.

3.1.2. Organizar el Equipo de trabajo

Dependiendo del tipo de aplicación selecciona y de la complejidad de esta se asignaran desarrolladores con conocimientos específicos para dar apoyo en distintas características de la aplicación. En general se recomendaría no superar las 8 personas incluyendo el cliente y el líder de equipo por cuestiones de organización y delegación de deberes. Es primordial que los desarrolladores se encuentren organizados en un espacio amplio, con buena iluminación y sin barreras intermedias que puedan dificultar la comunicación, de este modo todo el equipo está al tanto de lo que se va implementando además de proporcionar un ambiente sin tenciones. El cliente, como miembro del equipo, debe asistir a cada una de las reuniones y estar presente en la medida de lo posible para realizar pruebas de aceptación y verificar las entregas, de esta manera el equipo obtiene opiniones claras del usuario final. El equipo tendrá autonomía en cuanto a que desarrollar, como desarrollarlo y de qué manera asignar las tareas, este tipo de prácticas motiva el trabajo del equipo y permite que sean más productivos; sin embargo es necesario que se esté revisando si su desempeño sigue los lineamientos del proyecto.

3.1.3. Diseño de pruebas:

Una vez que las historias han sido descritas se procede a definir en estas mismas las pruebas unitarias básicas que deben aprobar antes de pasar a la siguiente tarea tal como se observa en la Figura 24. Establecer las pruebas para el código que aún no se ha implementado hace que los desarrolladores deban cambiar su punto de vista al del usuario, así la programación de las funcionalidades se orienta desde el

principio en cumplir con aquellos posibles escenarios a los que se enfrentaría el cliente desde la interfaz de usuario.

Las pruebas unitarias que se aplican a cada artefacto funcional que se va liberando contribuyen al aseguramiento de calidad además de hacer más sencillas las pruebas de integración y demás, se avanza desde lo más pequeño hasta hacer las pruebas de sistema.

Historia de Usuario EfiSenDroid	
HISTORIA N°:	**FECHA:**
USUARIO:	**ELABORADA POR:**
NOMBREDE HISTORIA:	
MODIFICACION A HISTORIA N°:	**VERSION:**
PRIORIDAD:	**RIESGO:**
DESCRIPCIÓN:	
PRUEBAS: • **Prueba 1** • **Prueba 2** • **….**	

Figura 24. Plantilla de diseño de pruebas en las historias de usuario.

Fuente: Elaboración propia.

Buscando que el procedimiento sea más organizado y sencillo de aplicar se establecen unos pequeños formatos para evidenciar la recolección de la información en cada etapa tal como se observa en la Figura 25.

Procedimiento EfiSenDroid Formato para datos Etapa 1 Definición			
Nombre del Proyecto	**Nombre del Cliente**		
Elaborado por	**Fecha (DD/MM/AAAA)**	**Versión**	**Iteración**

Resumen de la idea		
Objetivo Principal de la App		
Tipo de App	Nativa ☐ Web ☐ Hibrida ☐	
Dispositivo al que va dirigida	Smartphone ☐ Tablet ☐ Wear ☐ TV ☐ **Observaciones:**	
Miembros del equipo (Roles)		
Historias de Usuario de la Iteracion		

Figura 25. Formato FE1D para recolección de datos Etapa 1 Definición.

Fuente: Elaboración propia.

3.2. REPRESENTACION

En esta segunda etapa se organizara toda la información recogida en la primera, se tendrán en cuenta cada uno de los detalles charlados en la reunión de definición de idea y se crearan las tareas que tendrán como fin ir resolviendo cada uno de los requerimientos. Se harán bosquejos por cada historia de usuario y luego se integraran formando el Storyboard que será la guía del proyecto. Todas las tareas mostradas a continuación deben ser realizadas en conjunto con todo el equipo de

trabajo en especial con la supervisión del cliente. En la Figura 26 se observan las tareas pertinentes para la representación.

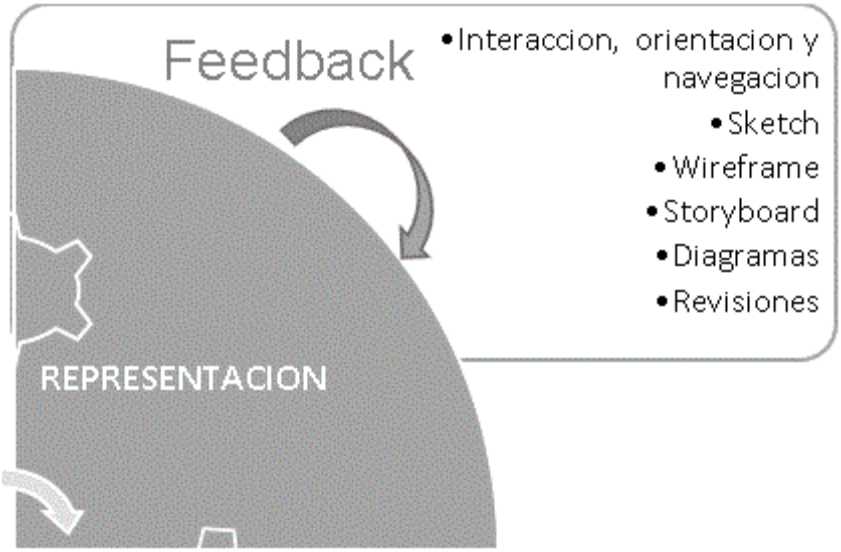

Figura 26. Tareas definidas para la fase de Representación.

Fuente: Elaboración propia.

3.2.1. Interacción con el dispositivo, orientación y navegación

Se deben tener en cuenta las formas en que los usuarios sujetan los dispositivos, cuales dedos tienen mayor interacción y como los usan, este análisis tiene incidencia directa en el diseño de la interfaz y condiciona la ubicación de los diferentes elementos interactivos en la pantalla. Habitualmente los usuarios sostienen, por ejemplo, el celular con una sola mano, lo cual les ofrece un poco más de libertad pero igual condiciona el uso de uno de los pulgares a la mayoría de las interacciones. Debido a esto se debe poner especial atención a las características anatómicas de la mano ya que así podremos determinar una mejor organización de los elementos de modo que sean fácilmente accesados. (Véase Figura 27).

Por otro lado, **Android** cuenta con algunas características propias de la interfaz del sistema que pueden ayudar a la selección de un diseño óptimo. Por ejemplo, cuenta

con una barra de navegación en la parte inferior para mayor accesibilidad, además esta cuenta con un botón atrás que permite regresar entre diferentes Actividades. Por otro lado presenta la opción de menús desplegables en la parte superior, lo cual es un buen ajuste si se tienen varias funcionalidades o varios grupos de contenido, además en el caso de aquellos botones que no se usan con frecuencia o que no deberían tocarse por error, como lo es el de eliminar o editar, se mantiene fuera del alcance, en Android por lo general en la parte superior.

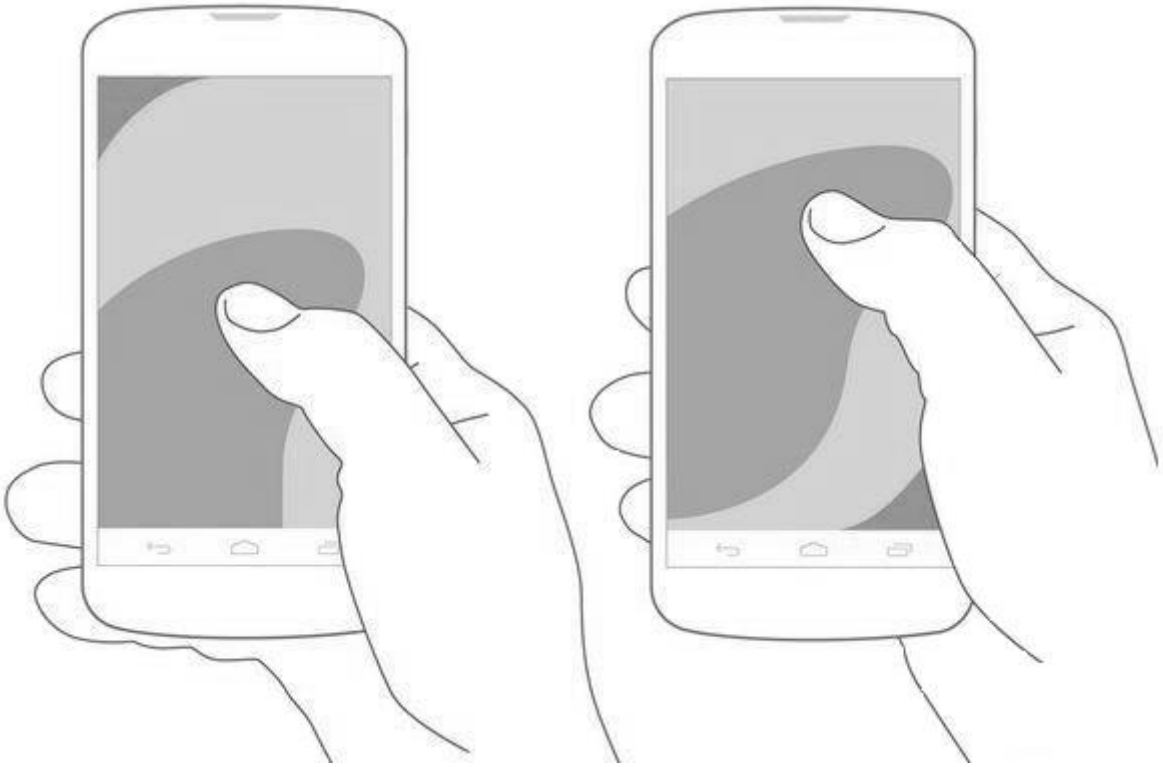

Figura 27. Accesibilidad del usuario sujetando el dispositivo con una mano.

Fuente: Cuello, J., Vittone, J. (2013). *Diseñando apps para móviles.* [Figura, ebook]. Disponible en https://play.google.com/

En esencia debemos acostumbrarnos a “Mostrarlo, no decirlo”, es decir hacer uso de los elementos visuales más que de las palabras, así orientamos todos los desarrollos a usuarios avanzados, ya que la interfaz será mucho más intuitiva y cualquier persona podrá descubrirla por si sola sin inconvenientes.

Aunque generalmente los celulares suelen tomarse manera vertical, dispositivos como Tablets cambian habitualmente las **orientaciones**, además existen aplicaciones que necesitan aprovechar la pantalla por lo cual cambian a modo apaisado (horizontal), por esta razón en recomendable diseñar teniendo en cuanta ambas orientaciones, sin embargo si la aplicación no hace necesario esto se puede implementar solo en el sentido relevante. La **navegación** debe ser simple y consistente, el usuario debe poder recorrer la aplicación sin necesidad de tutoriales, se debe seleccionar que es más necesario según el contenido, agregar solo las pantallas necesarias, no sobrecargar la interfaz, fácil acceso y regreso entre actividades. En Android se aconseja, si se hace uso de pestañas, usar de cinco a siete máximo.

3.2.2. Diseño Conceptual (Back-end)

Esta etapa, al igual que en cualquier otro proyecto de software, es de gran importancia para que el desarrollo del proyecto cumpla con los requisitos establecidos, tanto en alcance como en funcionalidad. Las tareas siguientes permiten definir con mucha más claridad lo que se quiere y se debe obtener para que el proyecto tenga éxito. Esta etapa permite aclarar los requerimientos del cliente desde su punto de vista y el del usuario final, el transcurso de esta etapa está expuesto a constantes cambios, es por eso que se debe tomar el tiempo suficiente y constante retroalimentación para que se evite incorporar o retirar funcionalidades en la etapa siguiente. El diseño principal se recreara a partir de los Sketch de las historias y se hará uso de Wireframes y Storyboard que servirán como plano a seguir a lo largo del desarrollo para el cliente y los desarrolladores.

3.2.2.1. Sketch (Bosquejos individuales)

Una vez que los requerimientos están organizados en historias de usuario cada uno de ellos tendrá un valor. Aquellos cuyo valor sea alto serán los

elegidos para generar los primeros bosquejos ya que pertenecen a requerimiento funcionales del sistema.

Una vez que este hecho el bosquejo será mucho más sencillo para el desarrollador llevarlo del papel al código. En el ejemplo de la Figura 28 se representa la visión del cliente con respecto a dicha funcionalidad, su desarrollo puede ser en una hoja de papel o haciendo uso de herramientas online. Lo ideal es que se plasmen las características más relevantes sin detalles personalizados.

Figura 28. Ejemplo de Sketch o bosquejo de historia.

Fuente: Adaptado de UIStencils. (s.f) Recuperado de http://www.uistencils.com

3.2.2.2. Wireframe y Storyboard

La Figura 29 muestra que los Wireframe se centran en "que hace, más no como se ve", son la hoja de ruta que ayudará a entender la conexión entre cada pantalla y cómo el usuario puede navegar a través de su aplicación.

Por medio de los Wireframe obtenemos una visión de cada interfaz (pantallas individuales) centrada en la asignación de espacio, tipos de contenido, funcionalidades, conexiones entre pantallas y comportamientos comunes. Su diseño es muy simple, sin colores para no distraer la atención con detalles estéticos y puede ser creada simplemente con lápiz sobre un papel o haciendo uso de aplicaciones para diseño.

Figura 29. Integración de Wireframes para construir el Storyboard.

Fuente: Tomado de *Wiki loves monuments mobile application.* (2016). Recuperado de https://www.mediawiki.org

La información que se representa en los Wireframes es verificada por el cliente y los desarrolladores, se deben identificar cuales historias son realmente esenciales y cuáles pueden ser descartadas. Por medio de estas representaciones se pueden ver posibles flujos de información y así establecer comportamientos en diferentes escenarios, es por eso que una vez hechos los Wireframes lo mejor es armar el Storyboard que mostrara

en secuencia cada una de las pantallas y el modo en que deberán interactuar.

3.2.2.3. Diagramas de arquitectura

Dependiendo de la complejidad de la aplicación que se quiera desarrollar será necesario la especificación de las estructuras de datos y procesos de control, si la aplicación tiene diferentes roles se deberán mostrar las funcionalidades accesibles por cada rol, etc. Se recomienda en estos casos hacer uso de los diagramas UML y de bases de datos. Sin embargo si la aplicación es “sencilla”, aunque en la actualidad el valor agregado está en hacerlas de esta forma, con la definición clara de las historias y el Storyboard será suficiente para planificar la implementación a partir de estos.

3.2.2.4. Limitaciones

Es necesario que se tenga en cuenta los dispositivos reales en los cuales podrían instalarse la aplicación, las pruebas también deben hacer uso de equipos con estas características para validar su desempeño. No todos los dispositivos cuentan con las mismas referencias de hardware y software por lo que es necesario implementar una aplicación atractiva pero con un diseño simple más no deficiente. En el caso de que la aplicación necesite invocar algún componente o servicio por medio de un **Intent**, como por ejemplo la cámara, puede que esto repercuta en la duración de la batería, detalles así deben analizarse antes de sobrecargar la App con características innecesarias.

3.2.2.5. Revisiones

Es necesario que tanto desarrolladores como clientes supervisen contantemente los flujos de la información a través de los bosquejos elaborados. La idea es establecer un diseño claro que en los posible no

deba ser modificado posteriormente, por ello es aconsejable en esta etapa hacer pruebas por parte de personas ajenas al grupo de desarrollo. Se recomienda que el Storyboard sea visto y revisado por amigos, familiares que no necesariamente deban ser expertos en la materia para obtener diferentes opiniones. Este tipo de práctica facilita descubrir algunos fallos o deficiencias en el modelo actual.

3.2.3. Diseño Visual (Font-end):

En esta sección es donde se les da vida al Storyboard y se empieza a generar el estilo de la aplicación (Véase Figura 30). Para escoger la mejor opción para nuestro producto basta con estudiar lo referente a la empresa o el cliente particular, el género de la aplicación y tipo, a quien va dirigida, etc.

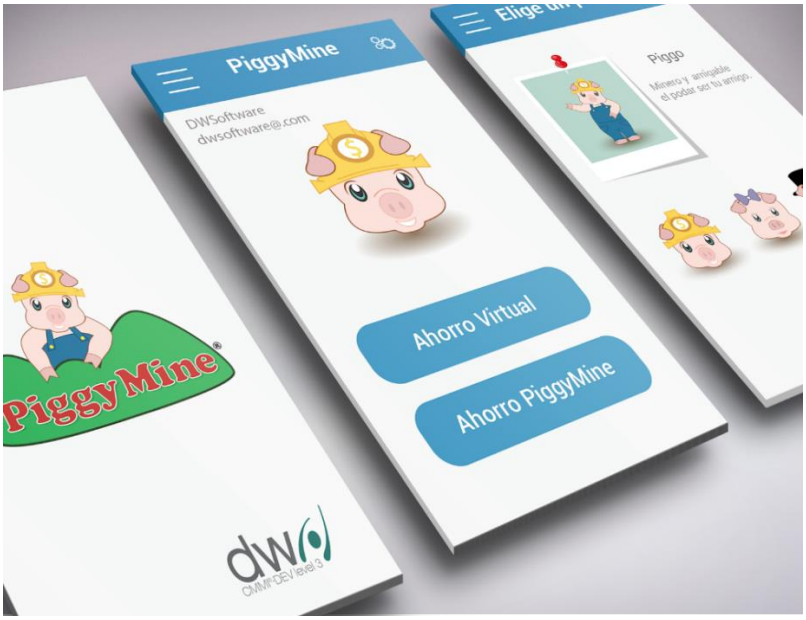

Figura30. Diseño visual, caracteriza la personalización de la interfaz.
Fuente: Contreras, B. (27 de Agosto 2015). *Principios para el diseño de aplicaciones móviles* [Figura en blog]. [Dwsoftware]. Recuperado de https://dwsoftware.mx

Si la aplicación es libre y no depende de factores como los anteriores Internet cuenta actualmente con un sin número de diseños que te permitirán escoger la mejor opción o te darán una idea de lo que posiblemente se esté buscando.

3.2.3.1. Pantalla de inicio e Icono

El principal atractivo de una aplicación vas más allá de su funcionalidad, hoy en día los usuarios prestan bastante atención a los detalles visuales, por tal motivo la pantalla de inicio y la selección de un icono para la aplicación generan gran impacto en cuanto a la primera impresión.

En general la pantalla de inicio se debería usar para mostrar la identidad de la empresa o el producto, ya que son los primeros objetos con los que interactúa el usuario, así que se hace publicidad cada vez que se abre la aplicación. Unas de las primeras tareas de diseño deberían ser la pantalla inicial y el icono, gastar algo de tiempo en recrear algo distintivo y llamativo pero a la vez sencillo ayudara a que la aplicación se destaque entre otras.

3.2.3.2. Estilo de interfaz

Una vez que se procede a agregar los objetos que harán parte de cada una de las actividades de la aplicación, nos encontramos con una estructura invisible la cual nos permite organizar los elementos visuales. Android cuenta con un módulo predefinido de 48 pixeles [21], algo así como unos 9mm, este es el tamaño mínimo recomendado para los elementos interactivos (botones, cajas de texto, iconos, etc.). La idea de manejar un módulo de este tipo es la de asegurar las dimensiones necesarias para que dichos elementos puedan ser tocados por el dedo sin problemas. En el caso del espaciado se recomienda un módulo de 8dpy los contenidos de fila de 4dp. Desde los bordes laterales hacia el contenido interno de 16dp. En la Figura 31 se observa la aplicación de los módulos para organizar el contenido en pantalla.

Figura 31. Módulos en Android para tamaños de elementos visuales.

Fuente: Cuello, J., Vittone, J. (2013). ***Diseñando apps para móviles***. [Figura, ebook]. Disponible en https://play.google.com/

3.2.3.3. Tipografía y color

Se aconseja la aplicación de Roboto (Véase Figura 32), la familia de fuentes tipográficas propias de Android, la cual es en gran parte la que da identidad al sistema junto con la combinación de estilo de botones y colores bien definidos. Estas características junto con la simplicidad del diseño llaman la atención de los usuarios, además permite que el texto se lea con claridad. Para lograr este resultado se debe tener en cuenta el tamaño, espacio entre líneas, ancho de columnas y colores de fondo, de este modo la información será visible sin importar las condiciones lumínicas del entorno. Se debe tener en cuenta detalles como la resolución de las pantallas de modo que no influya en la visualización del texto. La selección del tamaño de la fuente depende en gran medida del tamaño de los dispositivos y de la distancia al ojo del lector. En dispositivos con pantalla pequeña se debe seleccionar una fuente que simple y con separación de caracteres para aprovechar el espacio

disponible, Android maneja por defecto tamaños entre los 12 y 24 pixeles escalados, sin embargo el sistema proporciona la posibilidad de personalizar el tamaño en configuración.

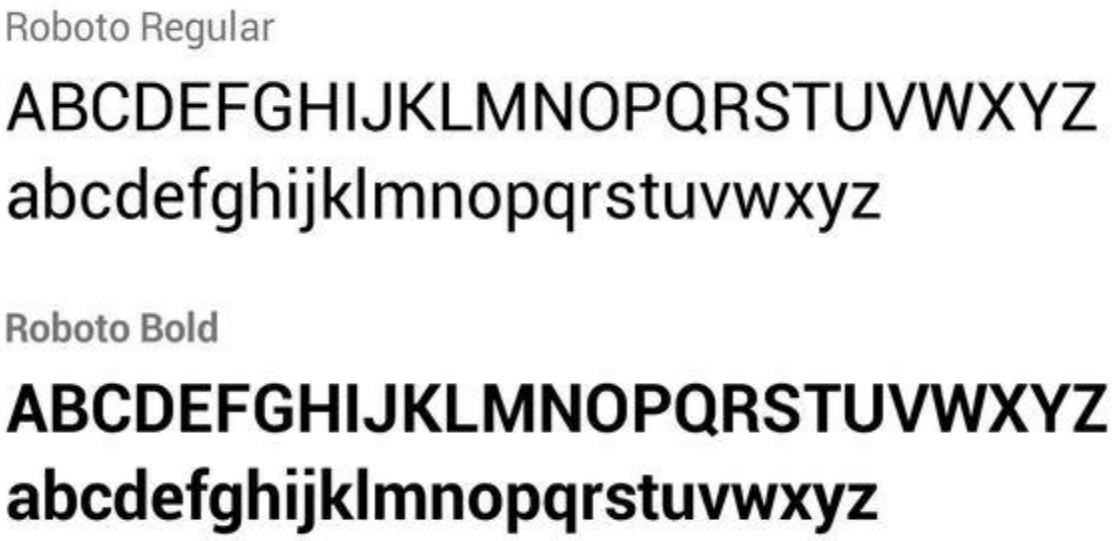

Figura 32. Roboto, fuente tipográfica oficial de Android basada en sans-serif.
Fuente: Elaboración propia.

Otro aspecto de vital importancia al definir el estilo de la aplicación es el color, la mejor manera de que resalte nuestra interfaz es por medio del manejo de escalas y contrastes, usar colores de manera sistemática de modo que tengan algún significado al usuario. Tener cuidado con el uso de colores como el rojo, amarillo y verde que son reservados para notificaciones de error, prevención o confirmación.

3.2.3.4. Idioma

Se debe prestar atención a las traducciones del contenido de la aplicación con se quiere presentar en diferentes idiomas, en ocasiones las palabras cambian sus tamaños al cambiar de dialecto lo cual puede afectar la correcta visualización del contenido. Se recomienda crear los directorios ***values*** necesarios para cada idioma dentro de la carpeta de recursos ***res/,*** identificarlos con el sufijo del idioma como se ve en la Figura 33, de

este modo Android cargara el recurso necesario según la configuración de idioma del dispositivo.

```
Myproject/
    res/
        values/
            strings.xml
        values-es/
            strings.xml
        values-fr/
            strings.xml
```

Figura 33. Creación de subdirectorios y archivos para localizar idiomas.
Fuente: Elaboración propia.

En la Figura 34 se presenta la plantilla que resume los datos involucrados en la etapa de representación.

Procedimiento EfiSenDroid Formato para datos Etapa 2 Representación			
Nombre del Proyecto	**Nombre del Cliente**		
Elaborado por	**Fecha (DD/MM/AAAA)**	**Versión**	**Iteración**
Orientación principal	Vertical ☐ Horizontal ☐ Ambas ☐		
Fuente tipográfica			
Color de texto			
Color principal de la interfaz			
Idioma de la App			
Historias de Usuario de la Iteración			

Figura 34. Formato FE2R para recolección de datos Etapa 2 Representación.

Fuente: Elaboración propia.

3.3. DESARROLLO

En este punto se inician las labores de recrear por medio de código cada uno de los aspectos que se definieron en el diseño. Cada proyecto se desarrolla de manera diferente sin importar el tipo de aplicación que se haya seleccionado. Por lo general las **apps nativas** serán las que más tiempo tomen en desarrollarse pero por su parte van a ser las que más posibilidades de personalización tendrán. Las **webapp o hibridas** son más sencillas por lo tanto su tiempo de construcción será mucho menor. En la Figura 35 se observan las tareas necesarias en la etapa de desarrollo.

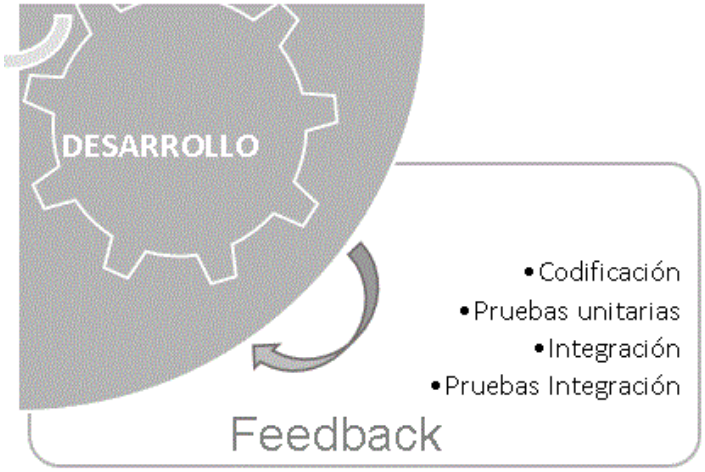

Figura 35. Tareas definidas para la fase de Desarrollo.

Fuente: Elaboración propia.

Existen muchos IDE's profesionales usados en la actualidad para desarrollar estas apps, es el caso de Android Studio un entorno integrado creado oficialmente por Google para desarrolladores, el cual incluye nuevas características por encima de Google App Inventor. En el pasado Google ya había implementado **plugins** para desarrollar en otras plataformas como por ejemplo Eclipse y Netbeans pero después

del lanzamiento de Studio se terminó el soporte para estos. Programación del código correspondiente para Actividades y Eventos

3.3.1. Programación del código correspondiente para Actividades y Eventos

En esta etapa las historias de usuario que se convirtieron en prototipo son codificadas por los desarrolladores, el proceso arroja una pantalla o actividad funcional, sin embargo para aceptarla no solo es necesario codificarla se deben aplicar las pruebas diseñadas en las historias de usuario. La iteración finaliza cuando el artefacto ha aprobado correctamente las pruebas y se ha re factorizado el código, en este punto se pasa a la siguiente tarea e inicia nuevamente la iteración. Este desarrollo incremental la base de pruebas permite la construcción paso a paso del producto software final aportando buenos índices de calidad del código y funcionalidad. Además cada iteración arroja retroalimentación de cada una de las pruebas aplicadas que se tiene en cuenta para las siguientes iteraciones.

3.3.1.1. Pruebas Unitarias

Se definen en reunión entre el cliente y el equipo, se deben realizar a cada actividad implementada para medir su funcionalidad de acuerdo a los requerimientos descritos en la historia, será el usuario final quien apruebe su integración junto con el resto del sistema. De acuerdo a los resultados el cliente decide si es necesaria alguna modificación o si se puede seguir con la próxima iteración.

3.3.2. Integración

Cada vez que una pantalla esta lista se debe ir integrando junto con cada las demás de modo que el cliente pueda ir probando los flujos de datos hasta este punto. Diariamente puede haber varias integraciones puesto que los programadores trabajan en un mismo sistema pero a partir de varias versiones de acuerdo a la sección que se esté desarrollando.

3.3.2.1. Pruebas de integración

En este caso las pruebas se realizan una vez que los skin o diferentes módulos son integrados, es decir las pruebas se realizan para ver la interacción entre dos o más partes del sistema. Así se verifica que los componentes actúen bien en conjunto.

3.4. PRUEBAS

Una de las etapas más importantes en cualquier desarrollo software, se procede con la aplicación de diferentes tipos de pruebas de modo que se puedan abarcar todas las funcionalidades requeridas en el proyecto y aquellas que no fueron definidas en primera instancia. En la Figura 36 se observan las tareas necesarias en la etapa de pruebas.

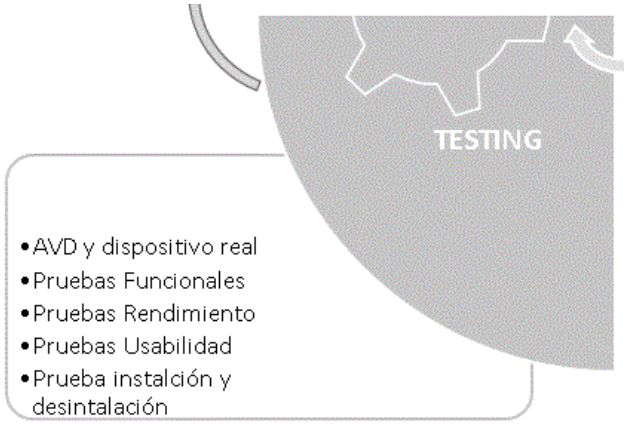

Figura 36. Tareas definidas para la fase de Pruebas.

Fuente: Elaboración propia.

Al igual que en otras etapas es necesario que tanto el cliente como el equipo de desarrollo este presente y atento a todas y cada una de estas pruebas de modo que al finalizar pueda ser liberado el producto con casi un 100% de certeza de que no hay deficiencias en tanto a usabilidad y funcionamiento. Cada una de estas pruebas deben ser ejecutadas tanto desde la terminal del emulador como desde un

dispositivo móvil físico, es de vital importancia ya que las pruebas de rendimiento deben usar los recursos de hardware para los que fueron creadas mas no los que se usaron para su desarrollo.

3.4.1. Emulador

Actualmente los entornos de desarrollo Android disponen de un administrador de AVD (Dispositivo Virtual Android) que permite la creación y administración de estos. Un AVD define las características de un teléfono o dispositivo Android de manera virtual, de este modo se pueden hacer pruebas del código como si fuera sobre el propio dispositivo a medida que se implementa. El AVD simula características como el tamaño de pantalla y la versión de Android del sistema, gracias a esto podemos ver resultados en tiempo real, la Figura 37 muestra un ejemplo de AVD en el IDE Android Studio.

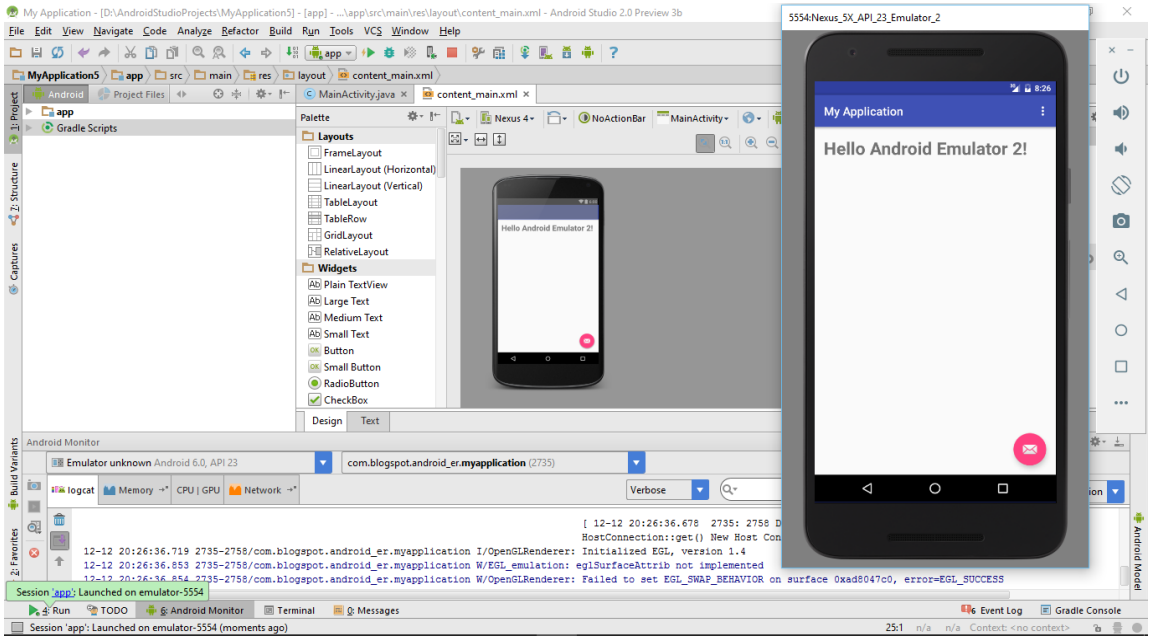

Figura 37. AVD del entorno de desarrollo Android Studio.

Fuente: Elaboración propia.

3.4.1.1. Pruebas Funcionales

En esta etapa se aplican pruebas que verifiquen que las funciones para las que fue desarrollado el software estén listas, por lo general a lo largo de las iteraciones se desarrollan de manera intrínseca en la búsqueda de terminar cada Actividad, sin embargo es necesario aplicar diferentes casos de control en determinados escenarios, lo ideal es plantear una prueba que amerite pasar por los flujos de información de la aplicación hasta terminar todo el proceso.

3.4.2. Dispositivo Físico

El uso del AVD proporciona gran comodidad a la hora de desarrollar diferentes aplicaciones, sin embargo es necesario que la aplicación sea empaquetada y que se pruebe directamente sobre un dispositivo real. Estas pruebas arrojan un verdadero valor a la calidad del producto final ya que se puede comprobar si las características del equipo permiten el buen desempeño tal como en el emulador.

3.4.2.1. Rendimiento

Se usa la aplicación tal como se hacía en el proceso de desarrollo, se hace uso de las funcionalidades que posee y se analiza su respuesta a cada tarea, tiempo de respuesta, fluidez, consumo de recursos en el dispositivo, etc.

3.4.2.2. Usabilidad

Unas de la pruebas más relevantes para proceder a su liberación, en estas es el usuario el que interactúa con la aplicación haciendo diferentes pruebas de las funcionalidades, el equipo de desarrollo se encarga de prestar atención a los resultados arrojados, un alto nivel de retroalimentación es recogido en esta prueba y permite analizar la satisfacción del cliente en cuanto al producto.

3.4.2.3. Instalación y desinstalación

Son pruebas que verifican el estado de la aplicación una vez que fue empaquetada en el archivo .apk, se verifica si la aplicación puede ser instalada sin ningún problema en el dispositivo y si puede ser desinstalada de igual forma. Se debe tener en cuenta que los dispositivos están configurados para no permitir aplicaciones de terceros, se debe cambiar esta opción en el apartado de configuración del equipo móvil antes de instalar.

3.4.2.4. Visualización

Es necesario que se analice la visualización de todos los elementos que componen cada actividad una vez que se ha instalado la aplicación en el dispositivo real, por lo general en el emulador los desarrolladores constantemente organizan cada elemento de manera armónica, sin embargo cuando se usan dispositivos reales la aplicación se expone a cambio en el tamaño de pantalla, recursos de hardware y software por lo que podrían cambiar también la manera en que fueron organizados.

En la Figura 38 se muestra la plantilla para la recolección de datos en la etapa de Pruebas.

Procedimiento EfiSenDroid Formato para datos Etapa 4 Pruebas			
Nombre del Proyecto	**Nombre del Cliente**		
Elaborado por	**Fecha (DD/MM/AAAA)**	**Versión**	**Iteración**
Pruebas en AVD (Emulador)			
Pruebas funcionales	Aprobó ☐ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:**		

Pruebas en dispositivo Real	
Pruebas de Rendimiento	Aprobó ☐ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:**
Pruebas de Usabilidad	Aprobó ☐ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:**
Pruebas de Instalación y desinstalación	Aprobó ☐ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:**
Pruebas de Visualización	Aprobó ☐ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:**

Figura 38. Formato FE4P para recolección de datos Etapa 4 Pruebas.

Fuente: Elaboración propia.

Capitulo IV: Prototipo App Control de Asistencia

A continuación se presenta los detalles sobre un prototipo para una aplicación para el control de asistencia de los profesores. En este caso particular se hizo énfasis en el desarrollo de un prototipo funcional que emule el requerimiento principal de llevar la asistencia de un grupo de estudiantes permitiendo acceder a la aplicación y exportando finalmente listados sobre asistencia.

4.1. DEFINICION

La Figura 39 muestra el formato para la primera etapa diligenciado con los datos correspondientes.

Procedimiento EfiSenDroid Formato para datos Etapa 1 Definición			
Nombre del Proyecto	**Nombre del Cliente**		
Prototipo Control de Asistencia	Autor del proyecto		
Elaborado por	**Fecha (DD/MM/AAAA)**	**Versión**	**Iteración**
Autor del proyecto	10 /10/2016	0	1
Resumen de la idea	La idea que motiva este desarrollo es la de crear el prototipo de una aplicación que permita llevar control de asistencia para un profesor. Debe permitir crear un listado de los alumnos pertenecientes a una materia y poder calificar si asistieron o no a clase permitiendo que se generen los listados en archivos de texto con la información mencionada.		

Objetivo Principal de la App	Llevar el control de asistencia de determinados grupos de estudiantes asignando un valor (SI/NO asistió) a cada uno.
Tipo de App	Nativa ☒ Web ☐ Hibrida ☐
Dispositivo al que va dirigida	Smartphone ☒ Tablet ☐ Wear ☐ TV ☐ **Observaciones:** Dispositivos principales Moto g1 y Moto g2. Pantallas de 4,5” y 5” respectivamente, almacenamiento interno de 8-16GB, memoria RAM 1GB.
Miembros del equipo (Roles)	Cliente, Equipo de desarrollo == Autor del proyecto

Figura 39. Diligenciamiento Formato FE1D.

Fuente: Elaboración propia.

4.1.1. Reunión y definición de la idea

4.1.1.1. Historias de usuario

Los requerimiento principales para el prototipo se establecen mediante historias de usuario que permitan organizar en primera instancia un panorama de lo que se va a construir, dejando evidencia de los requisitos. A continuación se muestran algunas de la Historias de usuario creadas para el desarrollo del prototipo.

Historia de Usuario EfiSenDroid	
HISTORIA N°: 002	**FECHA:** 27 DE AGOSTO 2016
USUARIO: Autor del proyecto	**ELABORADA POR:** Autor del proyecto
NOMBREDE HISTORIA: Agregar Estudiantes	
MODIFICACION A HISTORIA N°:	**VERSION:** 0
PRIORIDAD(ALTA-MEDIA-BAJA): Alta	**RIESGO(ALTO-MEDIO-BAJO):** Alto
DESCRIPCIÓN: Como cliente quiero poder agregar datos del estudiante como código, nombres y apellidos a la lista	

PRUEBAS:

Figura 40. Historia Agregar datos a estudiantes.

Fuente. Autor del proyecto

Historia de Usuario EfiSenDroid	
HISTORIA N°: 003	**FECHA:** 27 DE AGOSTO 2016
USUARIO: Autor del proyecto	**ELABORADA POR:** Autor del proyecto
NOMBREDE HISTORIA: Calificar la Asistencia	
MODIFICACION A HISTORIA N°:	**VERSION:** 0
PRIORIDAD(ALTA-MEDIA-BAJA): Alta	**RIESGO(ALTO-MEDIO-BAJO):** Alto
DESCRIPCIÓN: Como cliente quiero poder seleccionar sí asistió o no el estudiante	
PRUEBAS:	

Figura 41. Historia Calificar Asistencia.

Fuente. Autor del proyecto

Historia de Usuario EfiSenDroid	
HISTORIA N°: 006	**FECHA:** 27 DE AGOSTO 2016
USUARIO: Autor del proyecto	**ELABORADA POR:** Autor del proyecto
NOMBREDE HISTORIA: Exportar Lista	
MODIFICACION A HISTORIA N°:	**VERSION:** 0
PRIORIDAD(ALTA-MEDIA-BAJA): Alta	**RIESGO(ALTO-MEDIO-BAJO):** Medio
DESCRIPCIÓN: Como cliente quiero poder descargar la lista de la aplicación con toda la información a un documento de texto	
PRUEBAS:	

Figura 42. Historia Exportar Lista.

Fuente. Autor del proyecto

Historia de Usuario EfiSenDroid	
HISTORIA N°: 008	**FECHA:** 30 DE AGOSTO 2016
USUARIO: Autor del proyecto	**ELABORADA POR:** Autor del proyecto
NOMBREDE HISTORIA: Importar Lista	
MODIFICACION A HISTORIA N°:	**VERSION:** 0
PRIORIDAD(ALTA-MEDIA-BAJA): Alta	**RIESGO(ALTO-MEDIO-BAJO):** Alto
DESCRIPCIÓN: Como cliente quiero poder importar una lista de asistencia a la aplicación y poder visualizarla	

PRUEBAS:

Figura 43. Historia Importar Lista.

Fuente. Autor del proyecto

4.1.1.2. Establecer objetivo

El objetivo principal de la aplicación será el de llevar el control de asistencia de determinados grupos de estudiantes asignando un valor (SI/NO asistió) a cada uno, con la posibilidad de que toda esta información tomada en la aplicación pueda ser asignada a un documento de texto que la contenga, generado por la misma aplicación.

4.1.1.3. Tipo de App y dispositivo

La aplicación será de tipo nativa. Se deberán desarrollar funcionalidades específicas para cada requerimiento así como detalles de personalización. La aplicación podrá ser descargada e instalada directamente en los dispositivos. La aplicación se desarrollara para dispositivos celulares con sistema Android superior a las versiones 4.0. En el caso particular del prototipo hará uso de un dispositivo Smartphone con pantalla de 4.5” corriendo Android lollipop 5.1. (Véase Figura 44).

Figura 44. Moto G 2013.

Fuente: Elaboración propia.

4.1.2. Equipo de desarrollo

El equipo de desarrollo para el desarrollo del prototipo está conformado por una sola persona representando los demás roles.

4.1.3. Diseño de pruebas

En esta actividad describiremos las pruebas propuestas para algunas de las historias de usuario más relevantes.

Historia de Usuario EfiSenDroid	
HISTORIA N°: 002	**FECHA:** 27 DE AGOSTO 2016
USUARIO: Autor del proyecto	**ELABORADA POR:** Autor del proyecto
NOMBREDE HISTORIA: Agregar Estudiantes	
MODIFICACION A HISTORIA N°:	**VERSION:** 0
PRIORIDAD(ALTA-MEDIA-BAJA): Alta	**RIESGO(ALTO-MEDIO-BAJO):** Alto
DESCRIPCIÓN: Como cliente quiero poder agregar datos del estudiante como código, nombres y apellidos a la lista	
PRUEBAS: • Debe permitir agregar varios estudiantes • El campo de código debe ser de tipo numérico • Debe mostrar advertencia por campos vacíos	

Figura 45. Definición de pruebas para Historia Agregar Estudiantes.

Fuente: Elaboración propia.

Historia de Usuario EfiSenDroid	
HISTORIA N°: 003	**FECHA:** 27 DE AGOSTO 2016
USUARIO: Autor del proyecto	**ELABORADA POR:** Autor del proyecto
NOMBREDE HISTORIA: Calificar la Asistencia	
MODIFICACION A HISTORIA N°:	**VERSION:** 0
PRIORIDAD(ALTA-MEDIA-BAJA): Alta	**RIESGO(ALTO-MEDIO-BAJO):** Alto
DESCRIPCIÓN: Como cliente quiero poder seleccionar sí asistió o no el estudiante	
PRUEBAS: • Mostrar ventana para calificar asistencia	

• Debe permitir cambiar el valor de la asistencia • Se debe ver la calificación que se dio en la asistencia

Figura 46. Definición de pruebas para historia Calificar Asistencia.

Fuente. Autor del proyecto

Historia de Usuario EfiSenDroid	
HISTORIA N°: 006	**FECHA:** 27 DE AGOSTO 2016
USUARIO: Autor del proyecto	**ELABORADA POR:** Autor del proyecto
NOMBREDE HISTORIA: Exportar Lista	
MODIFICACION A HISTORIA N°:	**VERSION:** 0
PRIORIDAD(ALTA-MEDIA-BAJA): Alta	**RIESGO(ALTO-MEDIO-BAJO):** Medio
DESCRIPCIÓN: Como cliente quiero poder descargar la lista de la aplicación con toda la información a un documento de texto	
PRUEBAS: • El botón de exportar debe abrir la ventana para exportar • Debe pedir que se le asigne un nombre al archivo que se va a exportar • Debe mostrar advertencia si no se especifica nombre	

Figura 47. Definición de pruebas para historia exportar lista.

Fuente. Autor del proyecto

Historia de Usuario EfiSenDroid	
HISTORIA N°: 008	**FECHA:** 30 DE AGOSTO 2016
USUARIO: Autor del proyecto	**ELABORADA POR:** Autor del proyecto
NOMBREDE HISTORIA: Importar Lista	
MODIFICACION A HISTORIA N°:	**VERSION:** 0
PRIORIDAD(ALTA-MEDIA-BAJA): Alta	**RIESGO(ALTO-MEDIO-BAJO):** Alto
DESCRIPCIÓN: Como cliente quiero poder importar una lista de asistencia a la aplicación y poder visualizarla	
PRUEBAS: • Debe permitir seleccionar una lista desde el almacenamiento interno • Al cargar se deben ver los datos de la lista en la aplicación	

Figura 48. Definición de pruebas para historia Importar lista.

Fuente. Autor del proyecto

4.2. REPRESENTACION

El diseño de los bosquejos se hará a partir de una plantilla digital y haciendo uso de aplicaciones online. La Figura 49 muestra el formato con los datos de la segunda etapa debidamente diligenciados.

Procedimiento EfiSenDroid Formato para datos Etapa 2 Representación			
Nombre del Proyecto	**Nombre del Cliente**		
Prototipo Control de Asistencia	Autor del proyecto		
Elaborado por	**Fecha (DD/MM/AAAA)**	**Versión**	**Iteración**
Autor del proyecto	10/10/2016	0	1
Orientación principal	Vertical ☐ Horizontal ☐ Ambas ☒		
Fuente tipográfica	La fuente selecciona fue Sans Serif para toda la aplicación		
Color de texto	Blanco y Negro		
Color principal de la interfaz	Azul institucional Universidad de Pamplona. Código RGB (1, 82, 112)		
Idioma de la App	Español		

Figura 49. Diligenciamiento Formato FE2R.

Fuente: Elaboración propia.

4.2.1. Diseño Conceptual

4.2.1.1. Sketch

Se muestran los bosquejos de las historias más relevantes para el desarrollo del prototipo. Estos bocetos están sujetos a cambios en el transcurso del diseño o en la implementación, así como la incorporación de pantallas adicionales.

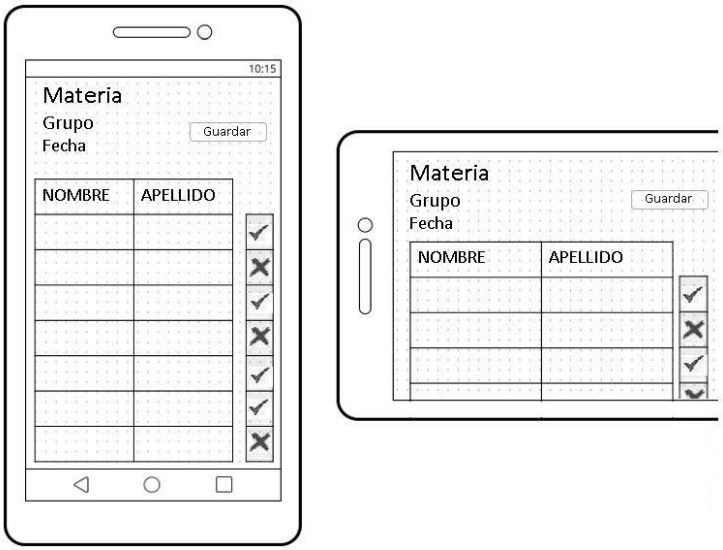

Figura 50. Sketch Historia de Usuario #003 Calificar Asistencia.
Fuente: Elaboración propia.

La Figura 50 describe la interfaz básica que se desplegara para controlar la asistencia, de esta manera el usuario deberá seleccionar de la lista con un toque si el estudiante asistió o no. En la Figura 51 vemos como debería verse la interfaz para la opción de cargar una lista de estudiantes que hubiera sido guardada previamente en el almacenamiento interno del dispositivo.

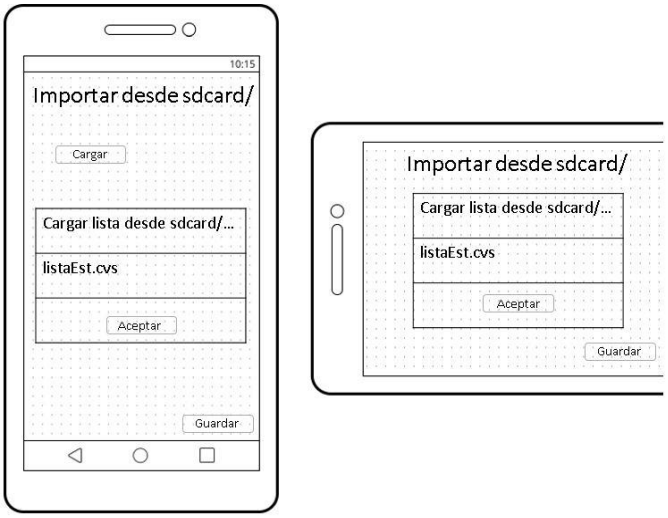

Figura 51. Sketch Historia de Usuario #008 Importar Lista.
Fuente: Elaboración propia.

A continuación, la Figura 52 muestra la interfaz para agregar las materias que contendrán los estudiantes según se estableció en la Historia Agregar Materias, este diseño se aplicara de igual forma a la de Agregar Grupos.

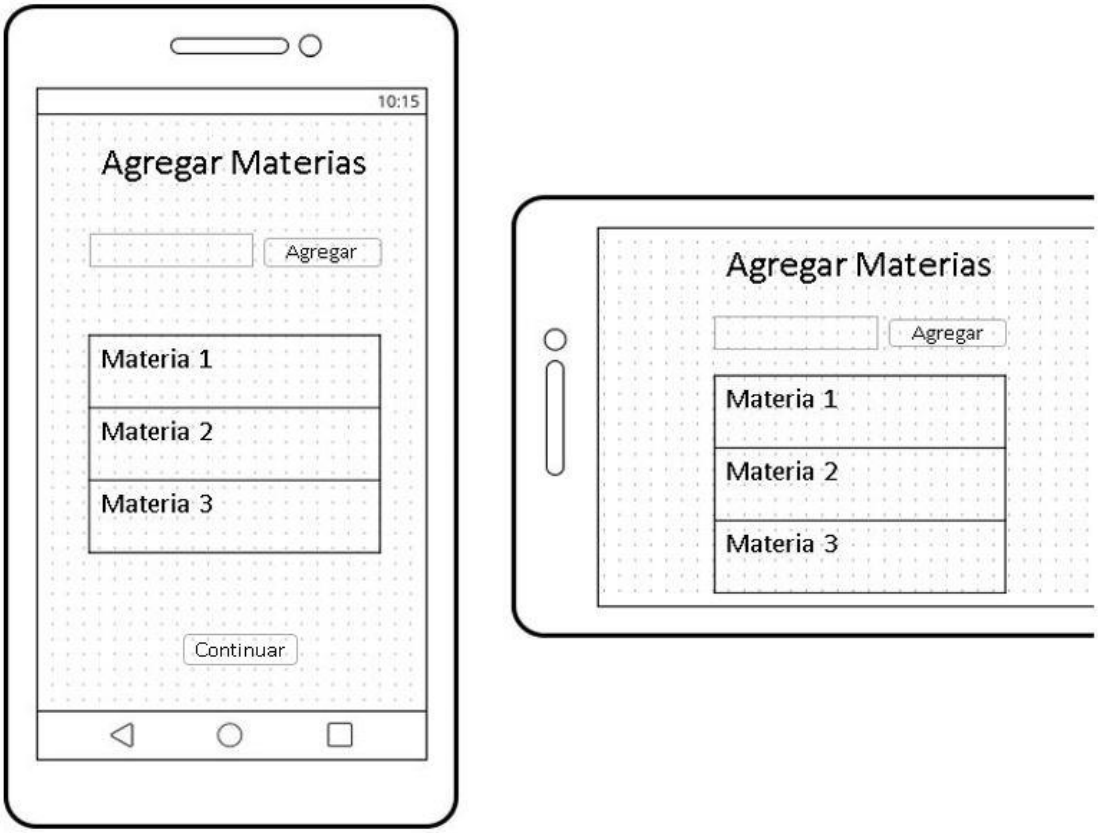

Figura 52. Sketch Historia de Usuario #013 Agregar Materias.
Fuente: Elaboración propia.

El bosquejo para de la Historia Agregar Estudiantes se muestra en la Figura 53, como podemos ver se establecen los campos más importantes para la identificación del estudiante, el código, nombres y apellido; en la Figura 54 se observa la interfaz para hacer el logín dentro de la aplicación, esta Actividad es necesaria por cuestiones de seguridad en cuanto a los datos que la aplicación debe manejar y almacenar, la idea es que una vez instalada se haga el proceso para almacenar las credenciales del usuario que tendrá único acceso.

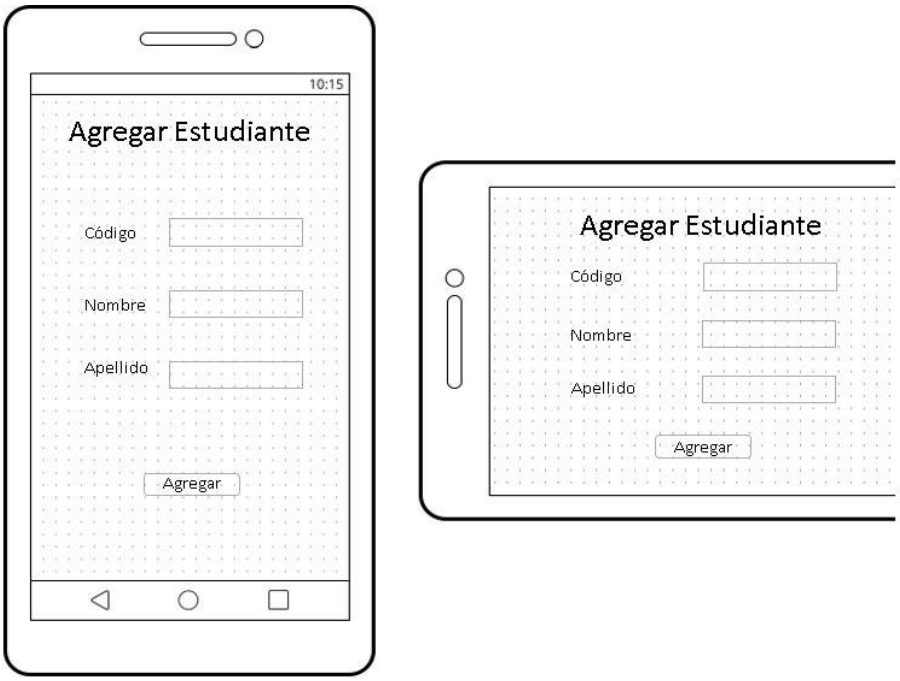

Figura 53. Sketch Historia de Usuario #001 Agregar Estudiantes.
Fuente: Elaboración propia.

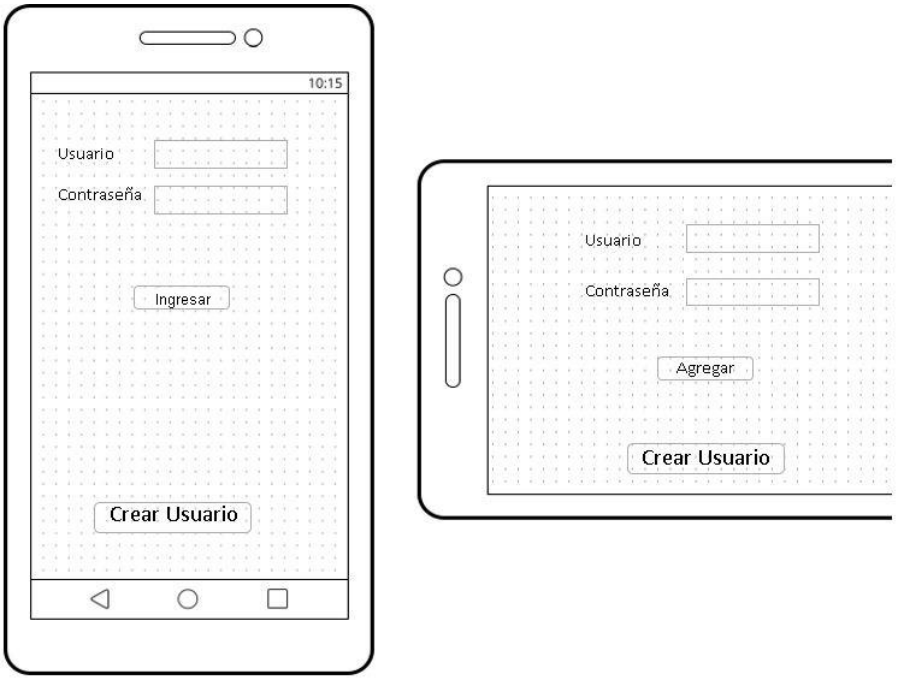

Figura 54. Sketch Historia de Usuario #012 Logín.
Fuente: Elaboración propia.

4.2.1.2. Wireframe y Storyboard

A continuación se muestran las interfaces diseñadas previamente pero caracterizando la navegación a través de estas, los Wireframes definen los comportamientos de los elementos en la interfaz, en conjunto todos los diseños muestran el Storyboard del prototipo y a partir de este se hará la implementación del código. La Figura 55 muestra los Eventos que se producen en la interfaz Logín antes de que se llame a una nueva Actividad, el botón atrás del dispositivo permite navegar también entre las diferentes Actividades de la aplicación.

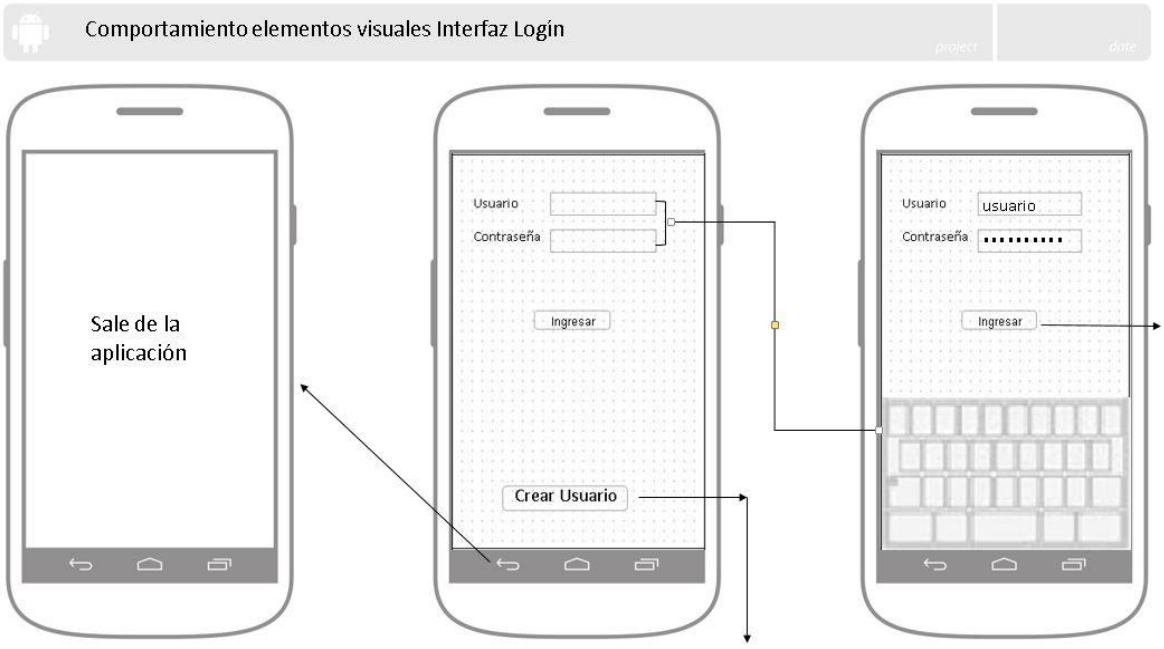

Figura 55. Wireframe Interfaz Logín.
Fuente: Elaboración propia.

En la Figura 56 se muestran los comportamientos de la interfaz de las Materias, esta misma definición de aplica en la navegación de la de los Grupos. En la Figura 57 se observa la navegación para la interfaz de Agregar estudiantes a determinado grupo y materia.

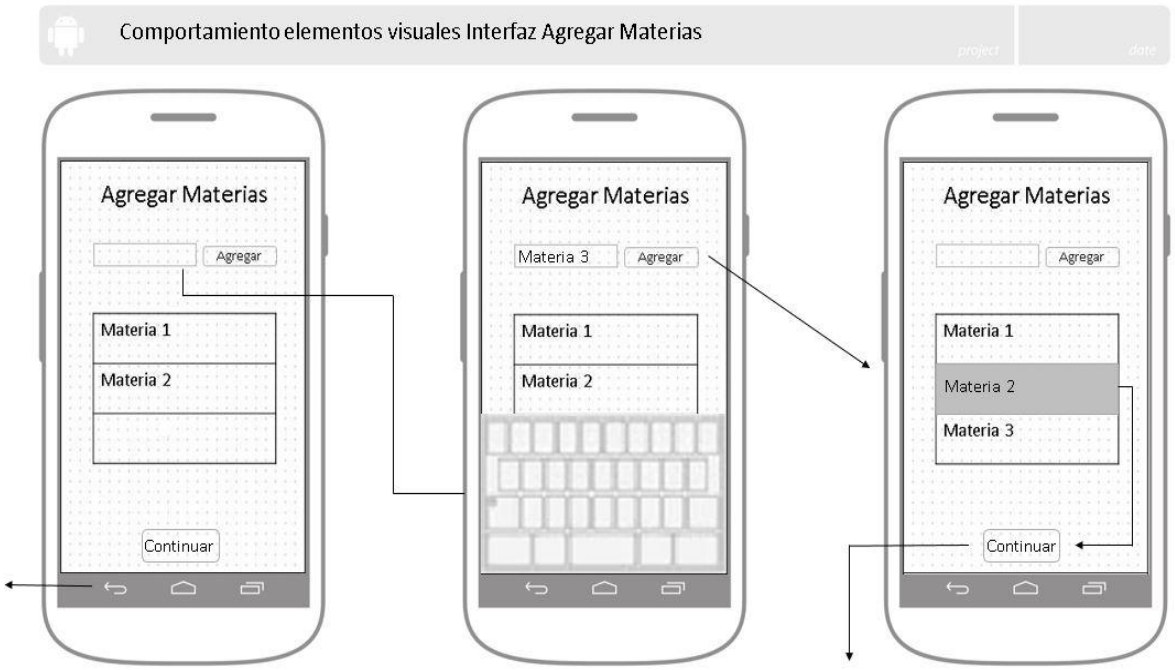

Figura 56. Wireframe Interfaz Agregar Materias.
Fuente: Elaboración propia.

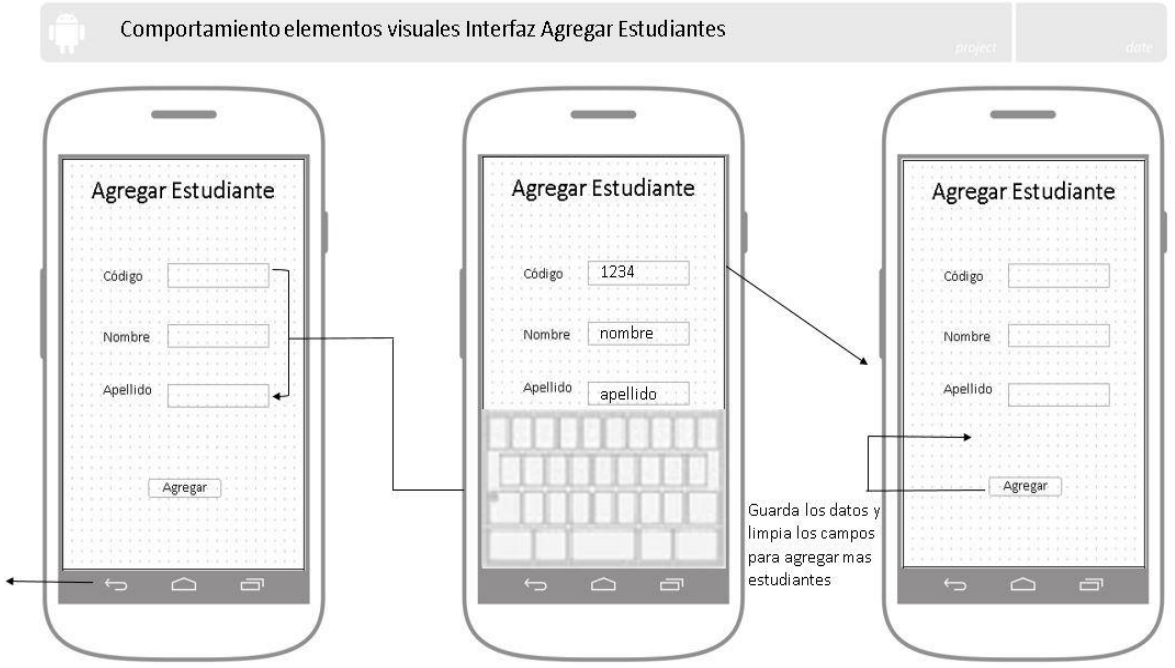

Figura 57. Wireframe Interfaz Agregar Estudiantes.
Fuente: Elaboración propia.

En la Figura 58 vemos los elementos visuales de la interfaz de Calificar Asistencia y sus comportamientos ideales.

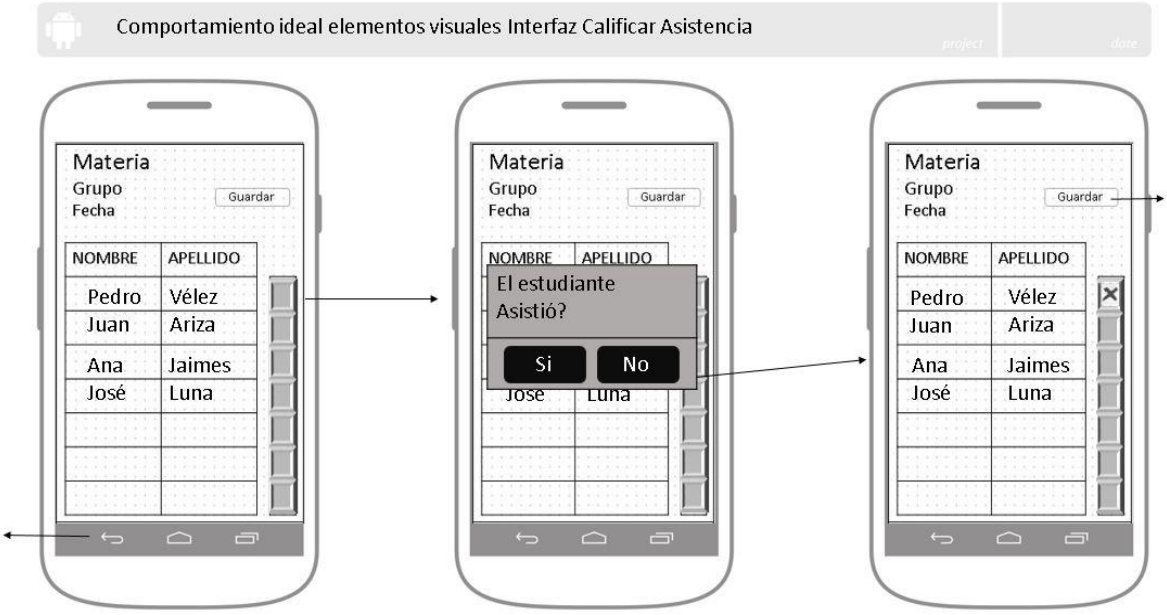

Figura 58. Wireframe Interfaz Calificar Asistencia.
Fuente: Elaboración propia.

Storyboard

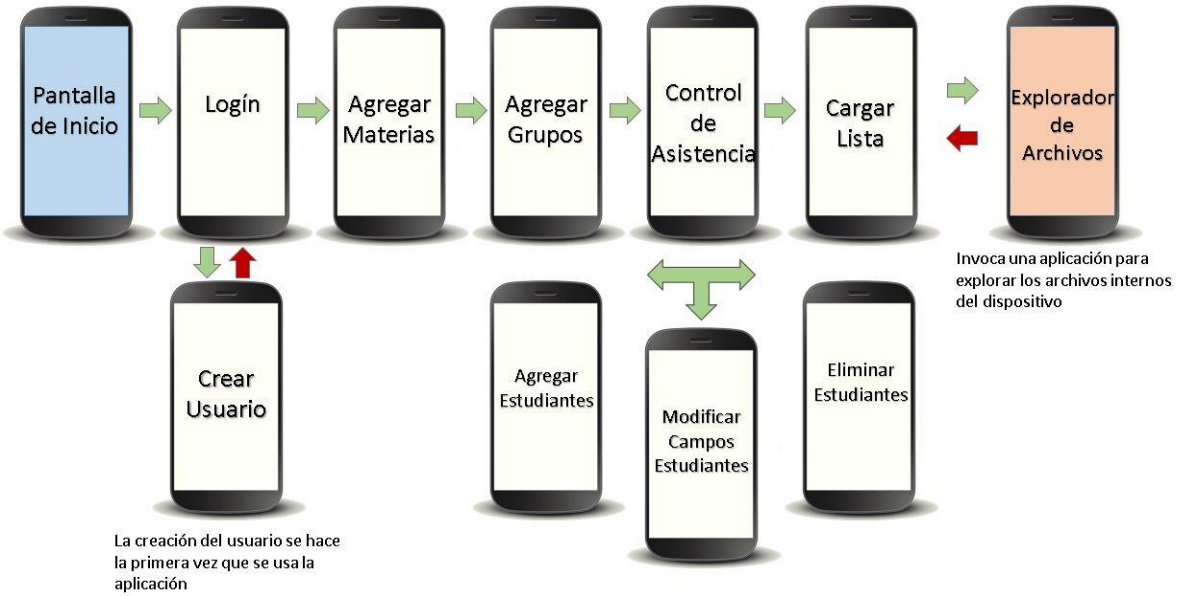

Figura 59. Storyboard general del Prototipo.
Fuente: Elaboración propia.

4.2.1.3. Diagramas de Arquitectura

Debido al que el prototipo no conlleva mucha complejidad y que existe un único rol no se hace necesaria la definición de diagramas adicionales, sin embargo, la Figura 60 describe las tablas que debería usar la base de datos (TinyDB) en por App Inventor 2 internamente.

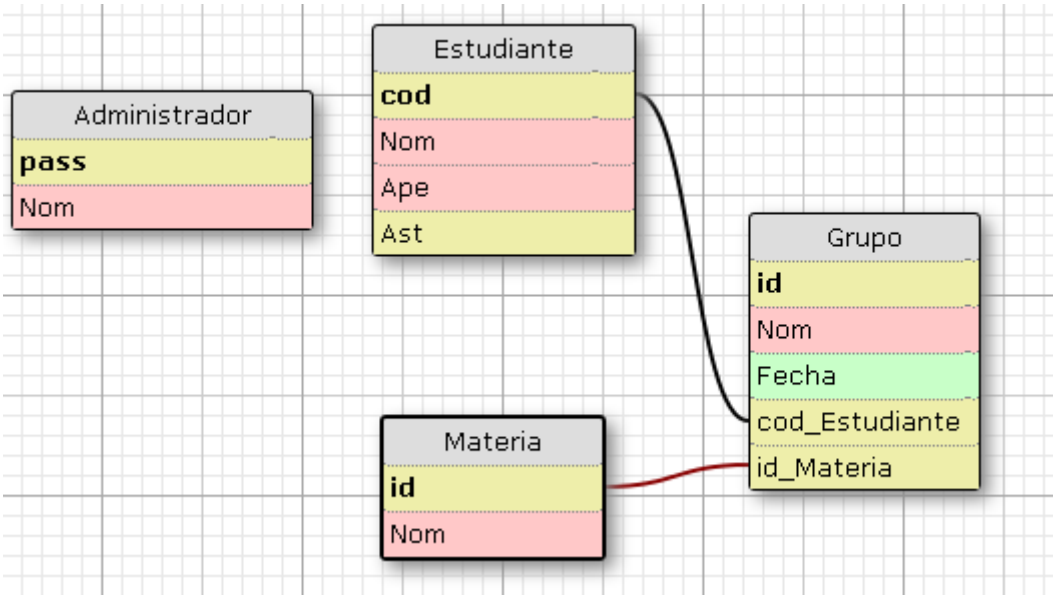

Figura 60. Tablas usadas en la base de datos.
Fuente: Elaboración propia.

4.2.1.4. Revisiones

A lo largo del proceso de diseño se hicieron las revisiones necesarias sobre aspectos visuales que deberían estar y la eliminación de algunos no indispensables, los pequeños cambios son evidentes en la etapa de implementación.

4.2.2. Diseño Visual (Front-end)

4.2.2.1. Pantalla inicial e icono

Se diseñó una interfaz simple en tanto a elementos visuales, se hizo énfasis en resaltar el nombre de la aplicación y en destacar el logo de la Universidad de Pamplona tal como se observa en la Figura 61.

Figura 61. Pantalla Inicial e Icono Aplicación Assistance Control.
Fuente: Elaboración propia.

4.2.2.2. Estilo de Interfaz

La interfaz se creó siguiendo los lineamientos recomendados para la plataforma Android, los tamaños de los elementos y su distribución se estructuraron según las características de la aplicación y la usabilidad que el usuario final le dará.

4.2.2.3. Tipografía y Colores

Se aplicó la fuente oficial en Android para lograr una buena visualización de los textos a lo largo de los elementos que la componen. La fuente en este caso fue **sans serif** base fundamental de Roboto. Algunos de los textos se adaptaron dependiendo de las características del dispositivo físico en que se iba a instalar, todo esto en busca de la buena presentación de los textos. Se seleccionó como

color fuerte el azul institucional tal como se ve en la Figura 62, en contraste con algunos grises para los elementos visuales y blanco para los títulos principales.

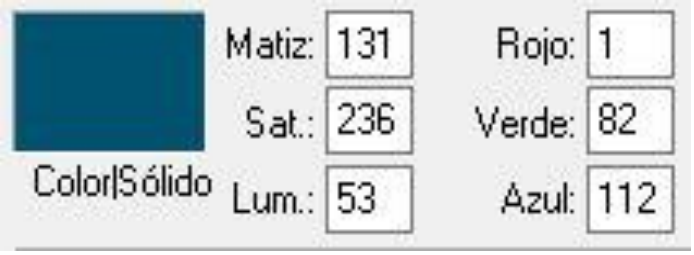

Figura 62. Selección de color principal en la aplicación.
Fuente: Elaboración propia.

4.2.2.4. Idioma

Lamentablemente debido al entorno usado para el desarrollo no fue posible configurar la aplicación en modo multi-lenguaje, se definió el idioma español por defecto para todo el prototipo.

4.3. DESARROLLO

4.3.1. Programación del código correspondiente para Actividades y Eventos

En este caso, debido al entorno de desarrollo seleccionado, la codificación de la aplicación se llevó acabo por medio de la librería de Open Block de Java, la cual permite la programación haciendo uso de bloques literales uniéndolos según la lógica de programación. El compilador se encarga de transformar estos bloques a lenguaje reconocible por Android por medio de lenguaje Kawa. En las Figuras 63,64 se observa el código correspondiente para la interfaz de Logín, en este caso App Inventor 2 presenta una pestaña para Diseño (elementos visuales) y otra para los bloques (programación de la parte visual).

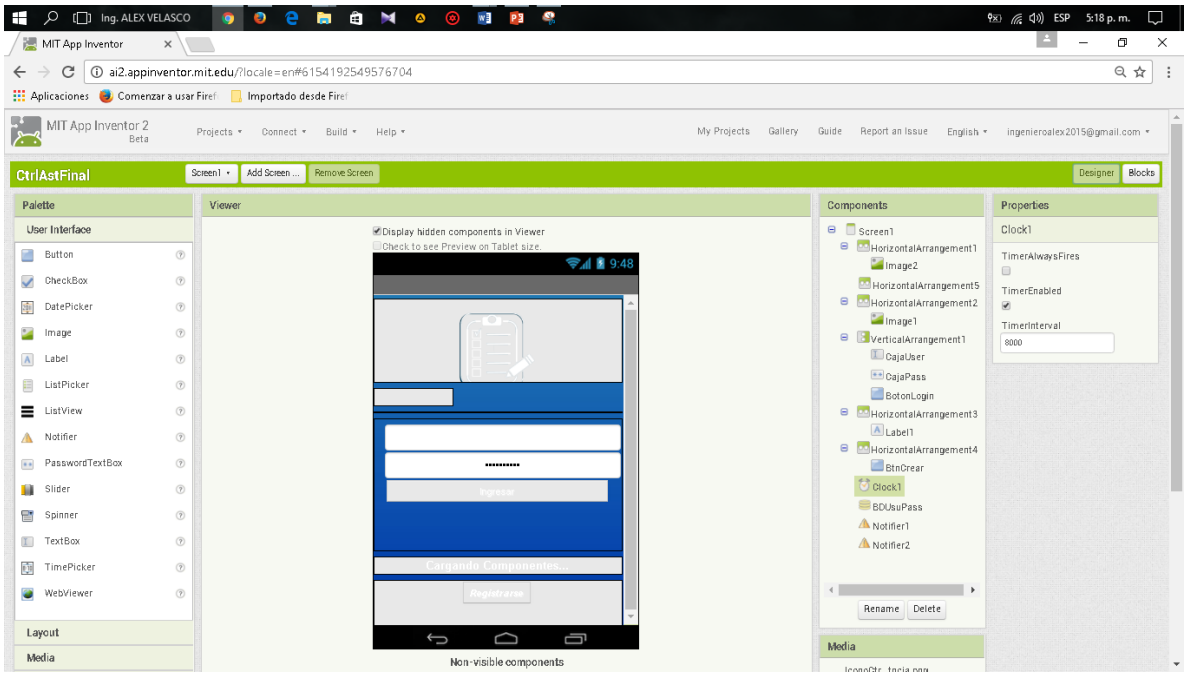

Figura 63. Programación interfaz Logín, vista de Diseño.
Fuente: Elaboración propia.

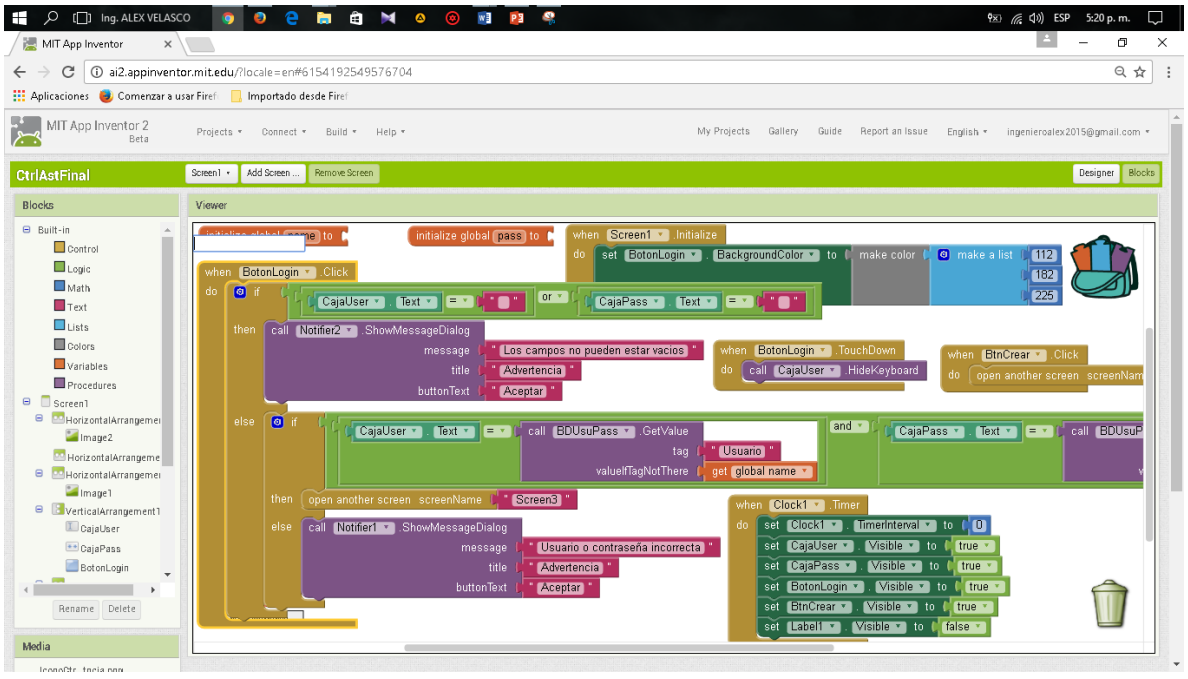

Figura 64. Programación interfaz Logín, vista de bloques.
Fuente: Elaboración propia.

A continuación se muestran las capturas del código de la interfaz central que califica la asistencia. (Véase Figuras 65-67).

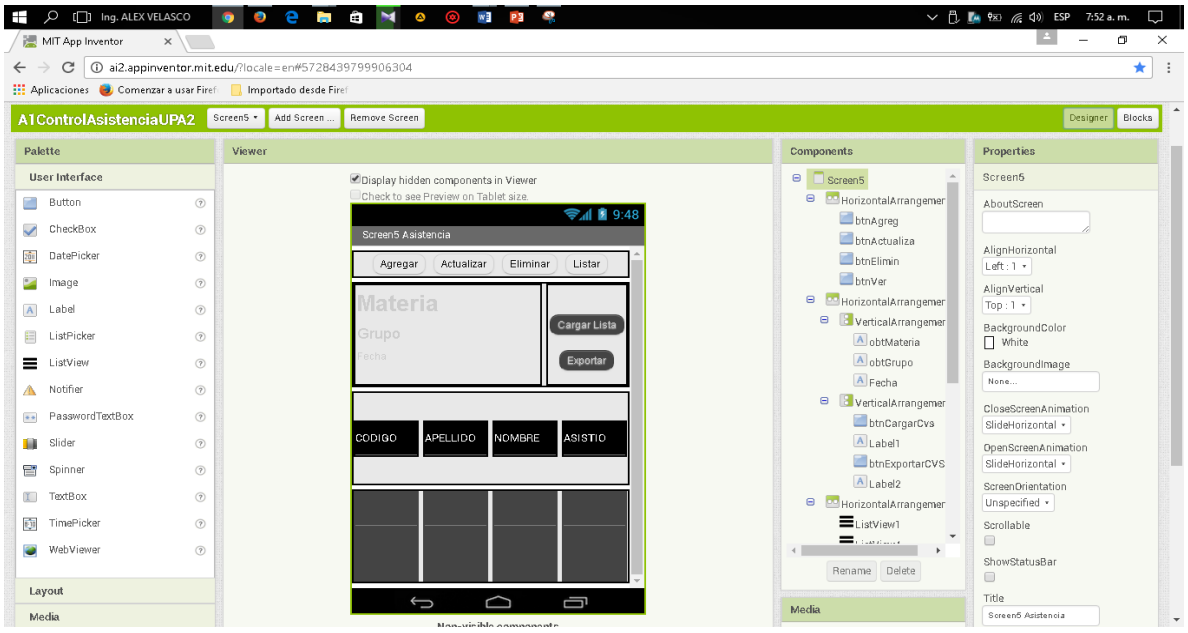

Figura 65. Programación interfaz Calificar Asistencia, vista diseño.
Fuente: Elaboración propia.

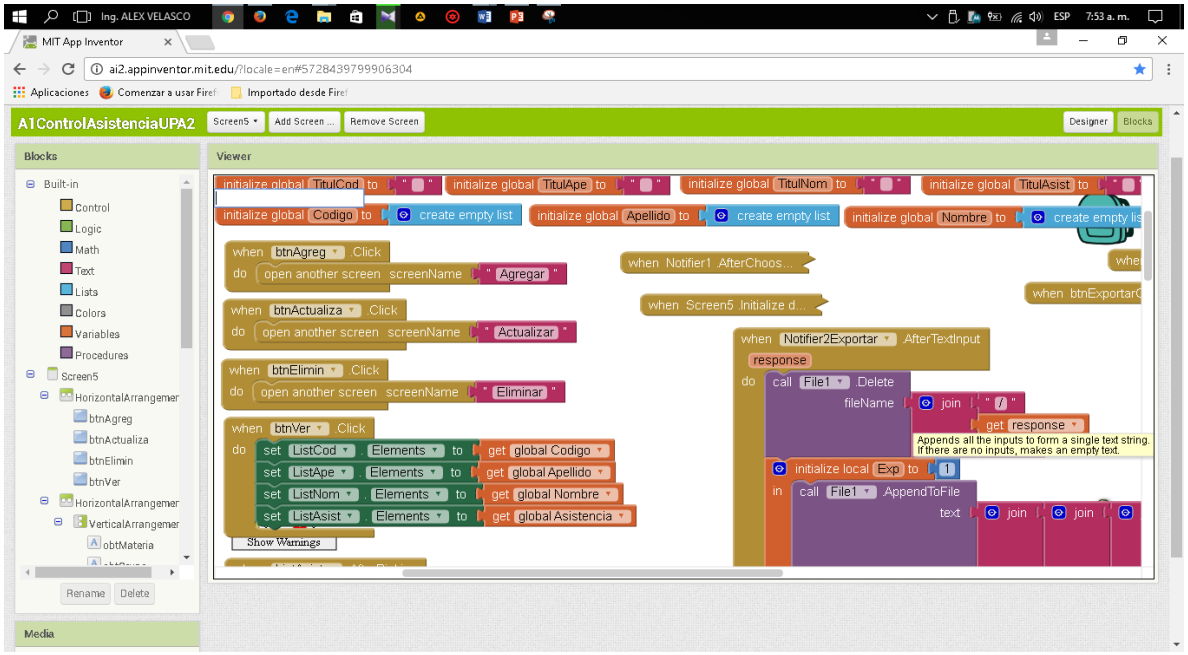

Figura 66. Programación interfaz Calificar Asistencia, vista bloques.
Fuente: Elaboración propia.

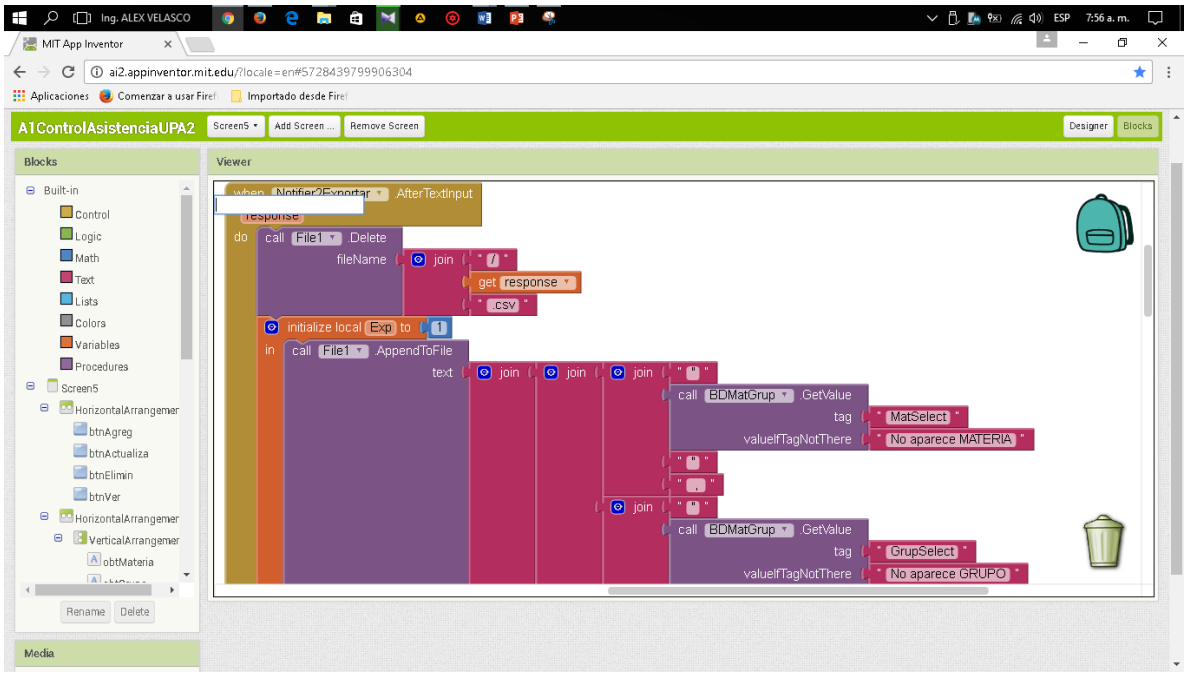

Figura 67. Programación interfaz Calificar Asistencia, vista bloques 2.
Fuente: Elaboración propia.

4.3.1.1. Pruebas Unitarias

Se aplicaron las pruebas diseñadas en las Historias de usuario de la mano con la implementación del código, la pruebas que fallaron se repitieron haciendo pequeños cambios en la codificación hasta que todas aprobaron. En las Figuras 68-71 se muestran algunas de las pruebas desarrolladas a lo largo del proceso de desarrollo.

Figura 68. Prueba los campos no pueden estar vacíos.
Fuente: Elaboración propia.

Figura 69. Prueba teclado numérico para campo código.
Fuente: Elaboración propia.

Figura 70. Prueba ventana de Advertencia usuario o contraseña incorrecta.
Fuente: Elaboración propia.

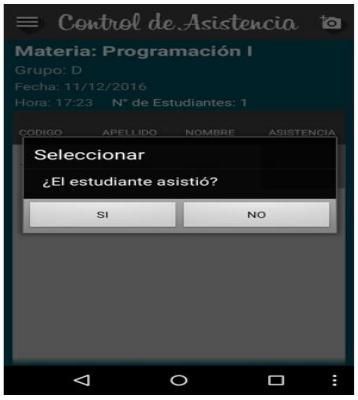

Figura 71. Prueba ventana calificar asistencia.
Fuente: Elaboración propia.

4.3.2. Integración

La integración de las diferentes Actividades es un proceso que se va dando a lo largo del proceso (Véase Figura 72,73), a medida que avanza la construcción del código los artefactos individuales se unen con otros para formar el producto final.

4.3.2.1. Pruebas de Integración

Figura 72. Prueba de integración de Actividades.
Fuente: Elaboración propia.

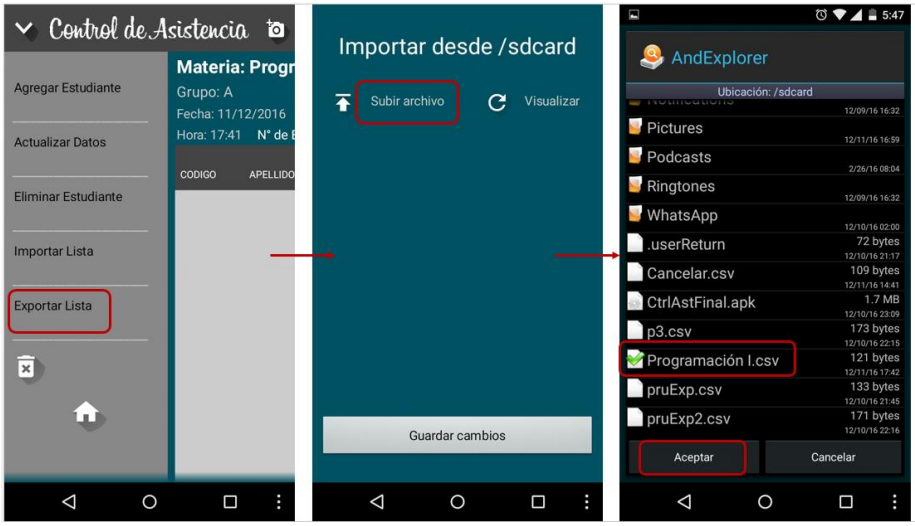

Figura 73. Prueba de integración de Actividades 2.
Fuente: Elaboración propia.

4.4. PRUEBAS

A continuación se evidencian las diferentes pruebas aplicadas al producto software en pro de su aceptación final por parte del cliente. En el formato definido para esta se etapa (Véase Figura 74) se observan algunas de las observaciones hechas en cada tarea.

Procedimiento EfiSenDroid Formato para datos Etapa 4 Pruebas			
Nombre del Proyecto	**Nombre del Cliente**		
Prototipo Control de Asistencia	Autor del proyecto		
Elaborado por	**Fecha (DD/MM/AAAA)**	**Versión**	**Iteración**
Autor del proyecto	12/10/2016	0	1
Pruebas en AVD (Emulador)			
Pruebas funcionales	Aprobó ☐ Falló ☐ Aprobó haciendo cambios ☒ **Observaciones:** Las pruebas transcurrieron sin problema, fueron aplicadas a las diferentes funcionalidades y fueron aprobadas. Se hicieron pequeños cambios en el transcurso para reparar algunos detalles visuales.		
Pruebas en dispositivo Real			
Pruebas de Rendimiento	Aprobó ☒ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:** El rendimiento en el dispositivo real fue bueno, se alcana a notar una leve diferencia en cuanto a emularlo en la pc pero en general fluye muy bien.		
Pruebas de Usabilidad	Aprobó ☒ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:** En general la distribución de los objetos visuales permite un buen uso, no se dificulta el alcance de los objetos haciendo uso de una sola mano.		
Pruebas de Instalación y desinstalación	Aprobó ☒ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:** Transcurrieron sin problema. Solo fue necesario configurar dentro del dispositivo la opción de software de terceros para que permitirá la instalación pero más allá de eso paso la prueba.		
Pruebas de Visualización	Aprobó ☒ Falló ☐ Aprobó haciendo cambios ☐ **Observaciones:**		

	Se aprecian algunas distorsiones con respecto a las pruebas emuladas, sin embargo en cuanto a la distribución general de los elementos todo funcionó de manera correcta, aun haciendo cambios de orientación en el dispositivo.

Figura 74. Diligenciamiento formato FE4P.
Fuente: Elaboración propia.

4.4.1. Emulador

4.4.1.1. Pruebas Funcionales

Las pruebas en el emulador se llevaron a cabo a lo largo de todo el proceso de implementación. En las siguientes Figuras 75,76 se detallan las pruebas del código siendo emuladas en el AVD.

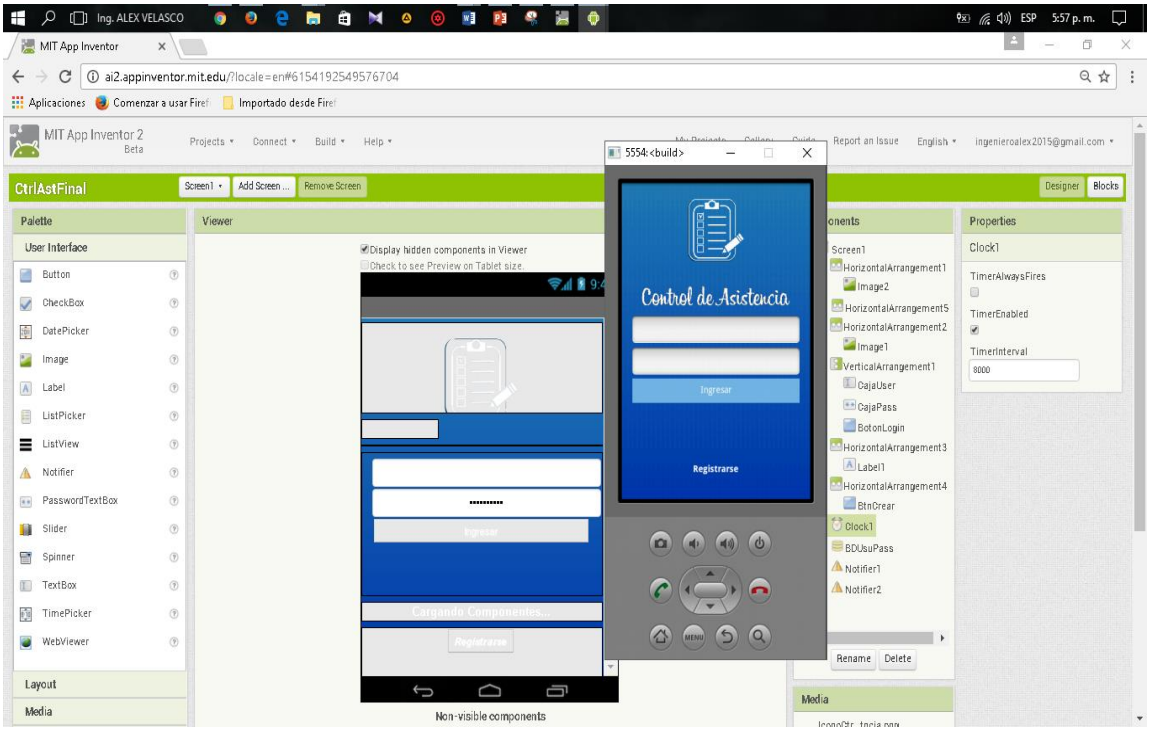

Figura 75. Prueba en simulador AVD.
Fuente: Elaboración propia.

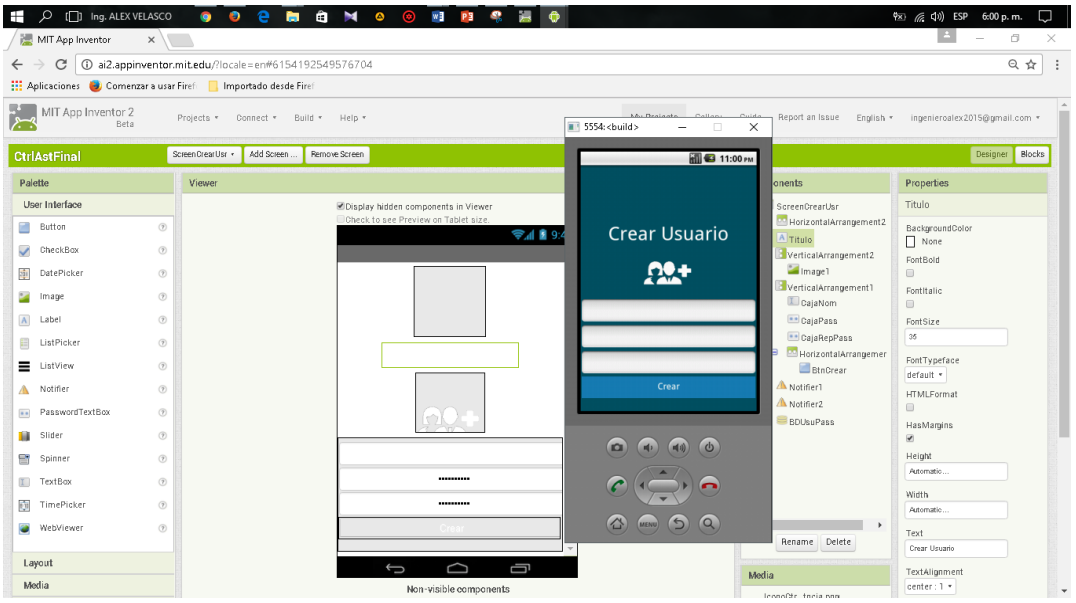

Figura 76. Prueba en simulador AVD 2.
Fuente: Elaboración propia.

4.4.2. Dispositivo Físico

Como se observa en el formato para recolección de datos de esta etapa uno de los dispositivos físicos usados fue el Motorola Moto G de segunda generación tal como se ve en la Figura 77.

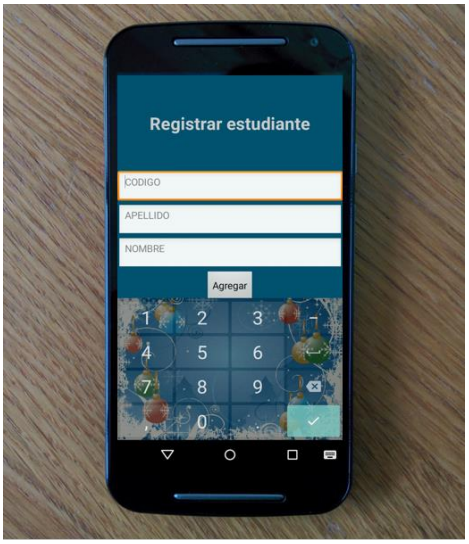

Figura 77. Prueba en dispositivo físico.
Fuente: Elaboración propia.

4.4.2.1. Rendimiento

En general la prueba de rendimiento avanzo sin problemas. El tiempo de arranque de la aplicación y las transiciones entre pantallas fueron aceptables. No se presentaron bloqueos de ningún tipo y se pudieron probar las diferentes funcionalidades de la aplicación.

4.4.2.2. Usabilidad

La usabilidad de la aplicación es buena, en términos generales cumple su función y es sencilla de asimilar. El acceso a los elementos está bien distribuido y se recoge buena retroalimentación de la aplicación de la prueba, toda esta información proporciona bases para mejoras en todos los aspectos del proceso de desarrollo.

4.4.2.3. Instalación y desinstalación

La aplicación se probó en diferentes dispositivos y no hubo dificultades con esta. El único detalle a tener en cuenta es la configuración que se debe hacer para que el dispositivo acepte software de terceros. En la Figura 78 se observan resultados de esta prueba.

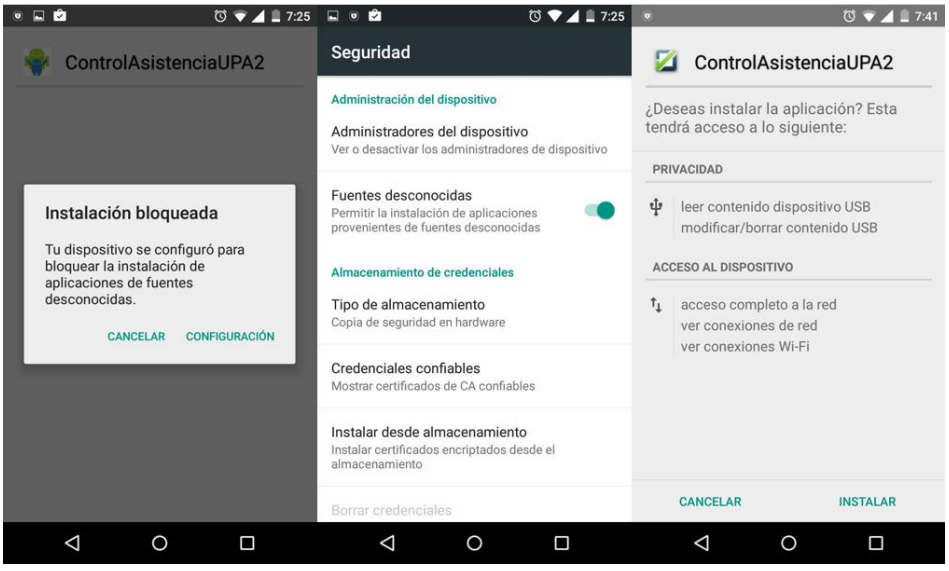

Figura 78. Prueba de instalación.
Fuente: Elaboración propia.

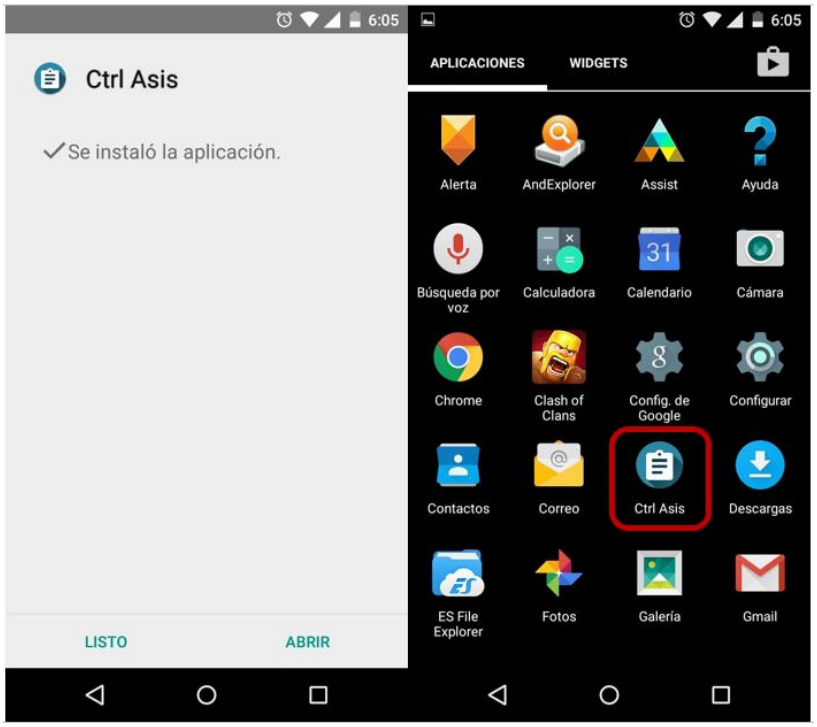

Figura 79. Prueba de instalación 2.
Fuente: Elaboración propia.

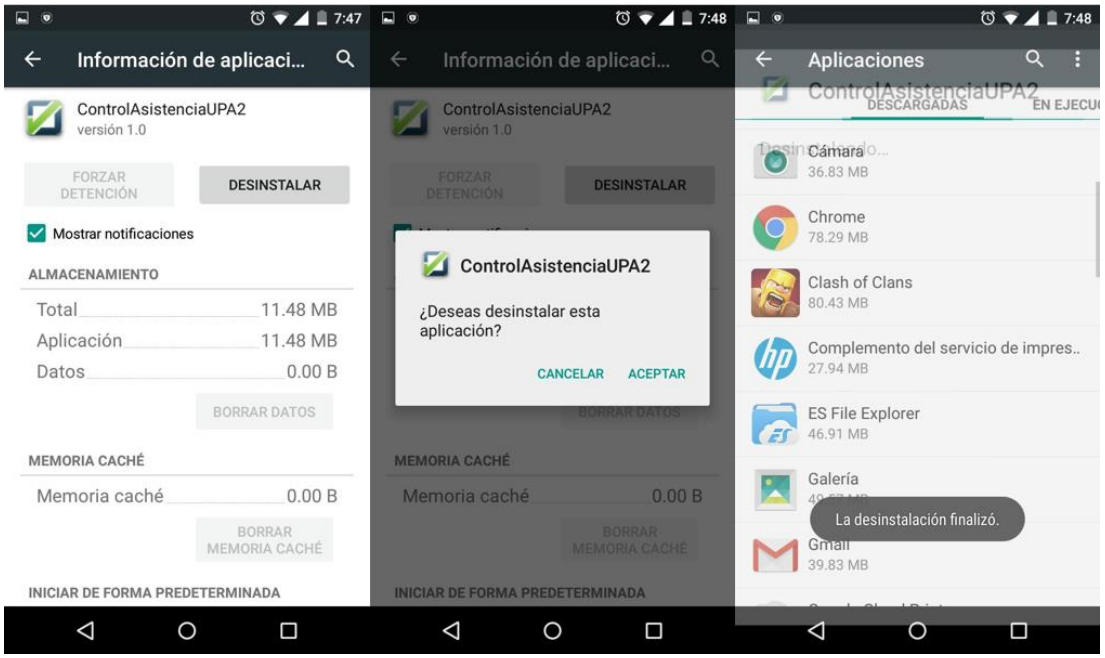

Figura 80. Prueba de desinstalación.
Fuente: Elaboración propia.

4.4.2.4. Visualización

La interfaz de la aplicación en general es buena, se apreciaron unas pequeñas alteraciones inicialmente pero pudieron ser superadas. La Figura 81 muestra las diferentes visualizaciones de una misma pantalla.

Figura 81. Prueba de visualización.
Fuente: Elaboración propia.

Capítulo V: Análisis de Resultados

La información recopilada en el trascurso de la investigación permitió sustentar la problemática tratada en este trabajo. Se confirmó la inexistencia de un procedimiento estándar que permitiera abordar proyectos con el objetivo de desarrollar aplicaciones móviles en la plataforma Android. Se evidencia también la necesidad de aplicar procedimientos y métodos tomados del paradigma ágil para la estructuración y seguimiento de implementaciones en el entorno móvil, puesto que poseen características únicas con respecto a otros desarrollos software y se desenvuelven en entornos con alta incertidumbre respecto al cambio.

Por medio del análisis a investigación previas se determinaron aquellas tareas agiles que arrojaban gran valor en la diferentes etapas del proceso de desarrollo, cada una de estas al incorporarse en la nueva propuesta junto con la atención a los detalles intrínsecos de los dispositivos móviles permitieron definir el procedimiento EfiSenDroid basado en la sencillez de su ejecución en ambientes cortos de tiempo y con poca estabilidad, así como la eficiencia para concretar un producto funcional al finalizar el proceso con un mínimo de recursos disponibles.

5.1. Caracterización del Procedimiento EfiSenDroid

El procedimiento propuesto, de acuerdo a lo que describen las tareas y actividades que lo conforman, tiene un enfoque en la identificación de características propias de los dispositivos móviles, el diseño de manera simple pero no deficiente de lo que será la aplicación y del aseguramiento de la calidad por medio de la ejecución de pruebas continuas en su desarrollo. De esta manera se asegura un cubrimiento de las etapas primordiales para la creación de cualquier producto software pero a la vez no se abusa de la sistematización de los procesos. En la Figura 12 comparamos la propuesta EfiSenDroid con los procedimientos y metodologías analizados previamente en cuanto a las características de sus etapas. Se definen los siguientes

parámetros: A. Proporciona soporte a la gestión del proyecto, B. Describe lo que se debe hacer en cada etapa, C. Define las actividades y tareas de la etapa.

Tabla 6.

Comparación de procedimientos con EfiSenDroid.

	Especificación de requisitos			Diseño			Codificación			Test Unitarios			Test de integración			Test de sistema			Test de aceptación		
	A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
SCRUM	X	X	X	X			X	X		X	X		X	X	X						
XP		X	X		X	X		X	X		X	X		X	X		X	X		X	X
MOBILE-D	X	X		X	X		X	X		X			X								
TDD					X	X		X	X		X	X		X	X		X	X		X	X
DRMOVIL	X	X		X	X	X	X		X	X											
AEGIS-MD	X	X	X				X	X					X			X	X		X	X	
MODELO E-HEALTH	X	X		X	X		X	X	X	X			X			X	X				
EFISENDROID	X	X	X	X	X	X	X		X	X	X		X	X		X	X		X	X	

Nota: Elaboración propia.

Podemos ver que gracias a las tareas que se realizan en cada etapa el procedimiento EfiSenDroid abarca todas las fases del desarrollo. Sin embargo, podemos decir que el principal inconveniente para la conformación del procedimiento fue la necesidad de verlo de manera general pese a que se intentaba enfocar a un uso específico para desarrollo en plataforma Android, es decir, aunque recoge características propias para abordar en aplicaciones del sistema Android no existe un desarrollo de aplicación fijo, cada vez que se selecciona una característica

para la App, como tipo, entorno de desarrollo a usar, finalidad de la aplicación, categoría, etc., hace que varié la forma en que se implementara.

5.1.1. Ventajas

- Organiza de manera general las tareas necesarias para el desarrollo de la aplicación.
- Mantiene un control de resultados por medio del uso de los formatos por etapa.
- Define recomendaciones sobre características propias de las aplicaciones en Android.
- Facilita el proceso de diseño y representación de los requerimientos.
- Sigue valores propios de los métodos agiles que facilitan la implementación en este tipo de proyectos.
- Hace énfasis en la aplicación continua de pruebas para la obtención de un producto de calidad.
- Se adapta y mejora por medio de la recolección constante de feedback en las tareas realizadas.

5.2. Aplicación del procedimiento para desarrollar el prototipo

El procedimiento cumplió los objetivos esperados, el uso de éste dio la estructura necesaria para definir la idea de desarrollo así como llevarla de la mano con el proceso de diseño e implementación. El resultado más palpable es el del prototipo funcional de la aplicación Control de Asistencia.

5.2.1. Criterios de evaluación

- **Sencillez:** Hace referencia a la facilidad de comprensión de las etapas del procedimiento y la manera en que se deben ejecutar.

- **Efectividad:** Representa la obtención del resultado software funcional con el mínimo de recursos.

Cabe resaltar que la etapa de Representación haciendo uso de Wireframes y Storyboard permite establecer de manera clara y muy visual los requerimientos dados por el cliente, lo cual facilita en gran medida la etapa de implementación en código de cada Actividad. Debido a las posibilidades de cambio en los requerimientos se hace muy importante el hecho de desarrollar cada pantalla por iteración, de este modo es mucho más sencillo y económico incluir cambios. Si es necesario incorporar modificaciones o nuevas funcionalidades basta con establecer la nueva Historia de usuario e incorporar el Wireframe dentro del Storyboard, con lo cual cada cambio incluye una iteración más en etapa de desarrollo.

Las tareas que permiten aclarar características propias de la aplicación Android a desarrollar favorecen la representación de los elementos visuales además de dar tip’s de acuerdo a que se debe tener en cuenta antes de abordar este tipo de proyectos. En este aspecto el procedimiento funciona como guía tratando no solo de estructurar el proceso de desarrollo sino también aportando recomendaciones que permitan prevenir ciertos aspectos que por lo general no se abordan y que pueden concluir en la generación de algunos fallos. Sin embargo, el procedimiento no es camisa de fuerza para la elaboración, debido a las características de este tipo de software en muchas ocasiones no se podrán seguir las mismas pautas, será necesario improvisar y hacer cambios de modo que adaptemos el desarrollo a nuestras necesidades. Por ejemplo, el desarrollo del prototipo no poseía gran complejidad en tanto a su implementación, por este motivo se obviaron algunas tareas como las de diagramación en UML que se recomienda en la sección de representación. Aun así son tareas que se sabe que son de gran utilidad cuando los proyectos son mucho más complejos y en ellos existen múltiples roles con diferente

tipo de acceso a las funcionalidades de la aplicación, por este motivo hacen parte de la definición del procedimiento pero no se hacen obligatorias.

Finalmente podemos decir que el procedimiento EfiSenDroid está basado en la sencillez gracias a que cada una de las etapas esta descrita por las actividades que se deben realizar y como realizarlas para estos ambientes cortos de tiempo y con poca estabilidad, así como en la eficiencia para concretar un producto funcional, en este caso el prototipo de la aplicación Control de Asistencia, al final del proceso con el mínimo de recursos aplicados.

5.3. Uso de formatos en etapas de desarrollo

Establecen una buena característica en cuanto a control y organización de las etapas. Permiten resumir los detalles relevantes de cada fase además de documentar el feedback recolectado en las diferentes iteraciones. Su aplicación a lo largo del desarrollo incrementa la posibilidad de mejorar en la siguiente etapa así como reducir los errores y la repetición de estos.

Capítulo VI: Conclusiones

El análisis de las fuentes consultadas y lo trabajos de referencia permitieron definir las bases necesarias para la construcción del procedimiento propuesto. Las diferentes propuestas encontradas en la actualidad ayudaron a determinar el área de oportunidad en la que esta propuesta seria relevante para la reducción de la brecha en cuanto a la formulación de un procedimiento oficial que abarque las características y problemáticas de los desarrollo de software móvil.

En cuanto al objetivo general planteado al inicio de la investigación podemos concluir que la definición de un procedimiento para el desarrollo de aplicaciones en Android fue logrado satisfactoriamente. De igual manera se logró la aplicación del procedimiento para el desarrollo del prototipo de aplicación en el cual se obtuvieron muy buenos resultados cumpliendo así con el objetivo de la propuesta.

Los resultados finales sugieren la necesidad de aplicar nuevamente el procedimiento variando la complejidad de la aplicación a desarrollar en pro de recolectar datos fundamentales para la consolidación del procedimiento. De este modo es posible analizar nuevamente las tareas definidas en diferentes entornos y con variabilidad de recursos, roles y tiempos.

REFERENCIAS BIBLIOGRAFICAS

Abrahamsson, P. (2004). *Mobile-D case studies homepage*. Recuperado de http://agile.vtt.fi/case.html

Abrahamsson, P. et al. (2005). *Mobile-D: an agile approach for mobile application*

Amaya, Y. (2015). *Guía metodológica ágil, para el desarrollo de aplicaciones móviles "AEGIS-MD".* Universidad Pedagógica y Tecnológica de Colombia. Revista de Investigaciones UNAD, Vol. 14. Recuperado de http://hemeroteca.unad.edu.co/

Babativa, A., Briceño, P., Nieto, C. & Salazar, O. (2015). *Desarrollo ágil de una aplicación para dispositivos móviles. Caso de estudio: Taxímetro móvil*. Universidad Distrital Francisco José de Caldas, Colombia.

Beck, k. (2000). *Extreme programming explained: embrace change.* Estados Unidos: Addison Wesley.

Beck, K. et al. (2001). *Manifesto for agile software development development, Conference on Object Oriented Programming Systems Languages and Applications*. Poster Session. Vancouver, Canada.

Beck, k. (2002). *Test Driven Development: By Example.* Estados Unidos: Addison Wesley Professional. ISBN: 0321146530.

Gasca, M., Camargo, L., Medina, B., (2013). *Metodología para el desarrollo de aplicaciones móviles*. Universidad de magdalena. Colombia, p. 24.

Hébuterne, S. y Pérochon, S. (2014). *Guía de desarrollo de aplicaciones para Smartphones y Tabletas*. (2ª ed.). Ediciones ENI.

IEEE *Standard for Developing a Software Project Life Cycle Process* 1074-2006.

Jacaboson, I., Booch, G. y Rumbaugh J. (2000). *El Proceso Unificado de Desarrollo de Software*. Massachusetts, Estados Unidos: Addison Wesley.

Kniberg, H. (2014). *Spotifyengineering culture (part1).* [Página web Laboratorios Spotify]. Recuperado de https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/

La Historia de Android. (2014). Recuperado de https://www.android.com/intln/es_es/history

Mafla, E. (2012). *Desarrollo de una aplicación para dispositivos móviles con sistema operativo Android aplicando Extreme Programming* (Tesis de pregrado). Universidad de las Américas. Quito, México.

Maigualca, M. (2012). *Análisis y estudio de las herramientas libres para el desarrollo de aplicaciones móviles en teléfonos celulares con sistema operativo Android, y la construcción de una aplicación para entretenimiento de usuarios en la carrera de informática y sistemas computacionales de la Universidad técnica de Cotopaxi.* (Tesis de pregrado). Universidad técnica de Cotopaxi. Latacunga, Ecuador.

Rahimian, V., Ramsin, R. (2008). *Designing and agile methodology for mobile software development: a hybrid method engineering approach*. Second International Conference on Research Challenges in Information Science. Marrakech.

Rodriguez, R., Prieto, A. y Sosa, E. (2011). *Programación Orientada a Objetos.* ISBN: 84-609-0003-7: Creative Commons.

Roque, R., Negrete, E., Salinas, J., (2013). *Aprendiendo a desarrollar aplicaciones para Android con la metodología agil Scrum: caso de estudio*, Mexico.

Sametinger, J. (1997). *Software engineering with reusable components.*

Schwaber, Ken. (2016). *Improving the Profession of Software Development.* Scrum.org. Recuperado de https://www.scrum.org/

Tipantasig, E. (2013). *Aplicación móvil utilizando plataforma Android para mejorar la calidad del servicio de consulta de información de consumo eléctrico de la EEASA en la empresa Besixplus CIA LTDA*. (Tesis de Grado). Universidad técnica de Ambato. Ambato, Ecuador.

Tumipamba, E. (2016). *Desarrollo de una aplicación movil que permite a los docentes y estudiantes de la Universidad Central del Ecuador acceder a las bases de datos cientificas* (Tesis de pregrado). Universidad Centra del Ecuador, Quito, Ecuador.

Vargas, D. (2014). Aplicación móvil para el seguimiento y control de las siembras de arrocera la esmeralda s.a. - Alesa Movil – Gaproa. Universidad del Valle. (Tesis de grado). Santiago de Cali, Colombia

Veloz, C. (2016). *Modelo para el desarrollo de aplicaciones nativas en Android basado en mejores prácticas, metodologías agiles y elementos del área interacción humano-computadora.* (Tesis de Maestría). Universidad Autónoma de Aguascalientes, Aguascalientes, México.

Zárate, H. (2008). *Paradigmas de Programación*. Universidad Nacional Autónoma de México. Ciudad de México, Mexico: Creative Commons

ANEXOS

Anexo A

Cultura de ingeniería de la empresa Spotify basada en Scrum. (Revisar carpeta de anexos en el CD).

Anexo B

Documentos en formato PDF de algunos de los procedimientos y metodologías tomadas como antecedentes. (Revisar carpeta de anexos en el CD).

Anexo C

Capturas de la codificación del prototipo CONTROL DE ASISTENCIA del presente proyecto. (Revisar carpeta de anexos en el CD).