Instituto Tecnológico Superior de Martínez de la Torre

Aplicación móvil para el monitoreo de productos avícolas

TESIS

PARA OBTENER EL GRADO:

Ingeniería en Sistemas Computacionales

PRESENTA:

Salathiel Méndez Sánchez

ASESOR:

MRYSI. Ángel Salas Martínez

Martínez de la Torre, Ver. MAYO, 2018

TECNOLÓGICO DE MARTÍNEZ	Nombre del Documento: **Carta de Autorización de Impresión**	No. Pág. 1/1

Nº. Oficio DET/ITSMT/ISC/031/2018

ASUNTO: Autorización de Impresión TESIS.

Martínez de la Torre, Ver., a 18 de Abril de 2018

C. **SALATHIEL MÉNDEZ SÁNCHEZ**
NO. DE CONTROL: 130I0153
EGRESADO(A) DE LA CARRERA
INGENIERÍA EN SISTEMAS COMPUTACIONALES
P R E S E N T E

Por medio de la presente hago constar que ha cumplido satisfactoriamente con lo estipulado por el Lineamiento para la Titulación Integral.

Por tal motivo **Se Autoriza** la impresión de la Tesis titulada:

"Aplicación móvil para el monitoreo de productos avícolas."

Dándose un plazo máximo de 30 días naturales a partir de la fecha de la expedición de la presente para realizar la solicitud del Acto de Recepción para la obtención del Título Profesional.

A T E N T A M E N T E

MRYSI. ÁNGEL SALAS MARTÍNEZ
Nombre y Firma
Presidente de Academia

MCA. YANAHUI BARRERA CASTELLANOS
Nombre y Firma
Jefe(a) de Carrera de Ingeniería

C.c.p. División de Estudios Profesionales.
C.c.p. Archivo

Índice general

Índice de figuras

Índice de cuadros

Agradecimientos

“No te rindas, debe de haber una solución”

Vegeta

En este presente trabajo de titulación agradezco especialmente a mi familia, quienes fueron los primeros en depositar su confianza en mí, además de darme las motivaciones para continuar y no darme por vencido, a mis amigos por los momentos compartidos y por ultimo extiendo un agradecimiento al Instituto Tecnológico Superior de Martínez de la Torre, principalmente a la academia de Ingeniería en Sistemas Computacionales, a los profesores que día a día me brindaron sus conocimientos en las aulas de clase, en especial al MRYSI Ángel Salas Martínez por proporcionarme su apoyo incondicional y brindarme la oportunidad de ser un profesional.

Resumen

En el desarrollo de esta investigación se exploraron dos de los sistemas operativos móviles más utilizados por los usuarios de Smartphone tocando temas como la historia de la plataforma Android, la evolución que esta ha tenido, además de las características de cada una de sus versiones. Con referencia a lo explicado anteriormente se analizaron dos de los entornos de desarrollo de aplicaciones móviles Android utilizados por los desarrolladores; se exploró la arquitectura que compone a esta plataforma, los elementos que componen la estructura de un proyecto *Android*, las fases de diseño y desarrollo que se emplearon para la creación del proyecto, así mismo se exploraron aplicaciones que existen actualmente en el mercado que cubren algunas de las necesidades solicitadas. Para el desarrollo de la aplicación se implementó la metodología ágil programación extrema la cual se enfoca en procedimientos cortos y rápidos, a lo largo de los planteamientos hechos se realizaron pruebas al producto final utilizando distintos dispositivos móviles, al concluir las pruebas se mostró que la aplicación permitió la captura de información para la actualización de inventario y su envió al servidor de la empresa, logrando obtener los resultados esperados en un entorno de trabajo real.

1. Introducción

En este mundo globalizado por la tecnología el uso de un dispositivo móvil ha llegado ser una necesidad indispensable para vivir, ya sea un *Smartphone* o una *Tablet*, estos dispositivos se pueden observar en cualquier ámbito como laboral o educativo, pues gracias a su fácil movilidad y acceso a internet ha sido fácil convertirse en las herramientas de trabajo de hoy en día. Dentro de las facilidades que nos brindan los Smartphone podemos encontrar el envío y recepción de mensajes mediante nuevas aplicaciones, el uso de video llamadas y el manejo de archivos de texto como son las paqueterías de office que actualmente ya se han incorporado.

Actualmente estos dispositivos móviles poseen un sistema operativo ya sea *IOS* o *Android*, con su incorporación a estos nuevos teléfonos inteligentes llegan consigo la introducción de las aplicaciones móviles o también conocidas como App(*application*). Hoy en día existen una extensa variedad de Apps que logran cubrir las diferentes necesidades de los usuarios con un *Smartphone* o *Tablet*.

La presente investigación tiene objetivo el desarrollo de una aplicación móvil para Smartphone que cuentan con sistema operativo Android, esto con la finalidad de hacer más eficiente el proceso de entregas, permitiendo que la información logre fluir de una manera más rápida manteniendo la información actualizada en tiempo y forma dentro de la empresa.

Para el desarrollo del proyecto se implementa el uso de la metodología ágil XP (Extreme programming), dicha metodología se caracteriza por enfocarse en procedimientos cortos y rápidos, adaptarse a cambios repentinos durante el desarrollo del proyecto, esto con la finalidad de lograr sus posibilidades de éxito. Además de la metodología de desarrollo se expone el tipo de investigación pura-aplicada la cual es empleada para el desarrollo de la investigación, se define el alcance que se tiene, así como las herramientas que se utilizan para la recolección de datos que brindan apoyo para corroborar el cumplimiento de las hipótesis planteadas al inicio de la investigación.

1.1. Planteamiento del problema

La empresa denominada El Shaddai, encargada de la distribución de productos avícolas, actualmente se ha logrado posicionar como el principal distribuidor de la zona norte de la empresa El Calvario, este lugar se debe a que El Shaddai entiende las necesidades del mercado por lo que no solo ofrece calidad de producto, además incluye calidad en el servicio que se brinda a los clientes.

Actualmente, la empresa cuenta con diez vehículos que operan dentro del estado de Veracruz, sin embargo, unos de los mayores inconvenientes con las que cuenta dicha organización es el tiempo que le toma a los choferes regresar con la información que se les otorga al finalizar cada entrega asignada, debido a que se ha llegado a presentar un retraso registrado hasta de dos días, ocasionando que los procesos que dependen de las entregas se detengan y por procedente no se logre concretar el cierre de inventarios provocando retrasos significativos dentro de la empresa.

> *"Los operadores que reparten producto por parte de la empresa en la parte norte del estado de Veracruz han llegado a tener la necesidad de hospedarse en esta zona entre lapsos de uno a dos días debido a la larga distancia que deben de recorrer, esto a su vez propicia que las remisiones que tienen a su cargo no logren ser entregadas en su debido tiempo, el retraso de las entregas de estos documentos ocasiona que la información no logre ser capturadas dentro de la base de datos de la empresa, ocasionando que esta información no esté disponible para poder ser consultadas, asimismo propicia que el inventario no logre ser cerrado en tiempo y forma".*

Pregunta de investigación

¿La implementación de una aplicación móvil brindará una mejor alternativa tecnológica para solventar la ineficiencia que existe en el proceso de captura de las devoluciones?

1.2. Hipótesis

Mediante la aplicación de un sistema web se llevará el control de los usuarios que se han registrado dentro de la aplicación.

Con la implementación de una aplicación móvil el operador capturará las piezas devueltas y el folio de la remisión, además reducir tiempo.

Con la utilización de una aplicación móvil el operador logrará hacer más eficiente el proceso de entregas.

Utilizando una aplicación móvil como medio de rastreo vehicular el personal de la empresa El Shaddai lograra conocer la ubicación actual de cada vehículo de reparto.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar una aplicación móvil para el monitoreo de pedidos y seguimiento de rutas de entregas avícolas para así llevar un control en tiempo y forma de las remisiones de las entregas y mantener la información en tiempo real.

1.3.2. Objetivo especifico

- Analizar las tecnologías para aplicaciones móviles.
- Generar una base de conocimiento debidamente documentada sobre el desarrollo de proyectos Android.
- Instalar los programas para el desarrollo de aplicaciones móviles Android
- Agilizar el proceso de Entregas de productos avícolas.
- Diseñar un apartado web que alimente la información que la aplicación móvil estará mostrando.
- Diseñar una interfaz de usuario que sea fácil de comprender y utilizar.

- Crear una conexión con la base de datos alojada en el servidor de la empresa y lograr visualizar la información dentro de la App.
- Autentificar a los usuarios de la aplicación.
- Realizar pruebas en diferentes tipos de dispositivos móviles.
- Determinar los requerimientos mínimos para que la aplicación se ejecute sin ningún problema en dispositivos basados en Android.

1.4. Justificación

Actualmente las empresas buscan la manera de agilizar sus procesos, reducir tiempo y mejorar la calidad de los servicios que ofrecen, es por ello que se busca la manera más eficiente de lograrlo, hoy en día existen tecnologías que están al alcance de grandes y pequeñas empresas. Por lo que El Shaddai no es la excepción y para ello pretende brindar un mejor servicio implementando el uso de una aplicación móvil para lograr agilizar sus procesos y manteniendo su información actualizada.

Tomando en cuenta la relevancia del uso de los dispositivos móviles, la presente tesis nace con el propósito de investigar, acerca del desarrollo de una aplicación móvil Android para agilizar el proceso de entregas dentro de la empresa El Shaddai, a consecuencia de que existe un gran lapso de tiempo muerto en el que se espera la recepción de la información necesaria por parte de los operadores, con la finalidad de poder apoyar a las próximas generaciones para servir de referencia si se desea desarrollar una App(*application*), que utilice el uso del GPS(sistema de posicionamiento global) del dispositivo móvil, conexión a base de datos en *SQL Server* y utilización del lenguaje *C#* como medio de conexión con *Android*.

1.5. Alcance y limitaciones

La presente investigación se enfoca en el desarrollo de una App(*application*) para dispositivos móviles que requiera la versión 5 del sistema operativo *Android* como mínimo.

Dentro de las ventajas competitivas, se ofrece privacidad de la información considerando que solo los usuarios con el rol de operador tendrán acceso a la lista de entregas por cumplir, lo cual se conseguirá mediante el uso de sesiones de usuario. En la primera versión de la App(*application*) se considera abarcar solamente lo que viene siendo el módulo de entregas, así como el seguimiento de rastreo GPS (sistema de posicionamiento global) de cada operador y el registro de las asistencias de cada usuario que tiene acceso a la aplicación.

1.6. Estructura del documento

Este trabajo se encuentra organizado en seis capítulos, con la intención de mostrar un panorama general del contenido del mismo, a continuación, se describe brevemente el contenido de cada uno.

El primer capítulo se expone la problemática con la que cuenta la empresa, los objetivos y el alcance que se pretenden lograr.

El segundo capítulo se narra la historia del sistema operativo Android, como surge y la evolución que ha ido teniendo al pasar de los años, se describe su arquitectura, los entornos de desarrollo más utilizados para la creación de aplicaciones móviles, fases para el desarrollo de App(*application*).

El tercer capítulo se exponen distintas aplicaciones móviles para la monitorización de vehículos.

El cuarto capítulo se expone el tipo de investigación, alcance, diseño, cronograma de actividades y finalmente los instrumentos de recolección empleados.

El quinto capítulo se lleva a cabo la recopilación y análisis de requerimientos para el desarrollo de la aplicación, se muestra los bosquejos que se realizaron antes de pasar la fase de implementación en el entorno de desarrollo, se muestra parte del código implementado para el desarrollo de la App(*application*), así como un diseño propuesto para la base de datos y finalmente pruebas de su implementación.

El sexto capítulo se habla expone los productos obtenidos al finalizar el desarrollo del proyecto y los resultados que se obtuvieron en su implementación.

El séptimo capítulo se explican las conclusiones obtenidas tras el desarrollo del proyecto, así como los trabajos futuros.

2. Marco teórico

2.1. Sistemas operativos para aplicaciones móviles

Es difícil definir qué es un sistema operativo aparte de decir que es el software que se ejecuta en modo *Kernel* (además de que esto no siempre es cierto). Parte del problema es que los sistemas operativos realizan dos funciones básicas que no están relacionadas: proporcionar a los programadores de aplicaciones (y a los programas de aplicaciones, naturalmente) un conjunto abstracto de recursos simples, en vez de los complejos conjuntos de hardware; y administrar estos recursos de hardware. [1]

Hasta ahora, los sistemas operativos que hemos utilizado fueron creados específicamente para las computadoras de escritorios o las computadoras portátiles, entre los más utilizados encontramos *Windows, Mac OS y Linux*, a diferencia de los dispositivos móviles como son *Smartphone* o tabletas son muy diferentes, es por ello que sus sistemas operativos son más simples.

En la actualidad el uso de sistemas operativos no se encuentra limitado únicamente al uso exclusivo de computadoras de escritorio o laptops, el constante desarrollo y evolución de estos SO (Sistema operativo) ha logrado que se pueda encontrar en dispositivos electrónicos que se ocupan comúnmente hoy en día, los Smartphone, tabletas, televisiones, etc. Son un claro ejemplo de productos en los que se puede apreciar su implementación,[2]define “sistema operativo El modulo software que controla la ejecución de los programas y que proporciona servicios tales como asignación de recursos, planificación, control de E/S y gestión de datos.

Los sistemas operativos móviles más representativos actualmente tenemos ha *IOS* desarrollado por la empresa *Apple* y *Android* desarrollado por *Android Inc.*, comprado por Google e impartido por este mismo.

Figura 2.1: logo *iOS* y *Android*

Fuente: [3]

iOS

iOS es el sistema operativo móvil que utiliza actualmente la empresa Apple en su Smartphone, al igual que en sus distintos productos, iOS originalmente se conoció como iPhone OS.

Versiones lanzadas:

iPhone OS

Fue la primera versión del sistema operativo móvil de *Apple*, se lanzó en junio 2007 junto con el *iPhone*, solo fue compatible con la primera generación de *iPhone* y *iPod Touch*.

Para marzo del 2008 se liberó el SDK (*Software Development Kit*) para desarrolladores, permitiendo poder crear *apps* para dicho dispositivo y se lanzó una actualización que permitía a los usuarios poder mover las *apps* en diferentes pantallas.

iPhone OS 2

Esta versión fue lanzada en junio del 2008, dicha versión vendría preinstalada para el *iPhone 3G*, las principales características en esta versión fueron:

- Implementación de la *Apps Store.*
- Mapas GPS (sistema de posicionamiento global).
- Correo electrónico *push*.

iPhone OS 3

Esta versión fe lanzada en junio del 2009, la llegada de la tercera versión del sistema operativo trajo consigo el estreno del *iPhone 3Gs*, dentro de las características fueron:

- Control por voz.
- Mensajería multimedia.
- Foco de búsqueda.
- Modo horizontal.
- Función copiar, pegar y cortar.

Figura 2.2: logos iPhone OS versión 1, 2 y 3
Fuente: [4]

iOS 4

Dicha versión fue lanzada en junio del 2010 junto con el *iPhone 4*, para el lanzamiento de esta versión *Steve Jobs* anuncia que se dejara de llamar *iPhone OS* y pasara a ser llamado como *iOS*, al igual que se anunció que se dejaría de dar soporte a las versiones más antiguas (primera generación), sus características fueron:

- Introducción de fondos de pantalla.
- Multitarea.
- *Face Time.*

iOS 5

Lanzada en octubre del 2011 ya integrado en el *iPhone 4s*, las características que introdujo esta versión fueron:

- *Siri.*
- Centro de notificaciones.
- *iMessage.*
- Redes sociales.
- *iCloud.*
- Actualizaciones *OTA* (elimino la actualización por cable).

iOS 6

Esta versión fue lanzada en septiembre del 2012 junto al *iPhone 5*, las características que trajo consigo fueron:

- La eliminación de los servicios de *Google*
- *Apple Maps*.
- Soporte LTE(*Long Term Evolution*)
- *PassBook*.
- Integración con *Facebook.*

Figura 2.3: logo de las versiones de IOS 4, 5 y 6

Fuente: [4]

iOS 7

Lanzado en junio del 2013 junto con el *iPhone 5s* y *5c*, se hizo un re diseño en la UI (*User interface*) de *iOS 7*, las características fueron:

- *iTunes Radio*.

- *CaPlay.*
- Nuevos iconos
- *Control center.*
- *AirDrop.*
- Nueva aplicación de fotos.

iOS 8

Lanzado en septiembre del 2014 junto con el *iPhone 6* y *iPhone 6 plus*, con el lanzamiento de esta versión se corrigieron errores que trajo consigo *IOS 7*, sus características fueron:

- *Apple Pay.*
- *HandOff.*
- *QuickType.*
- *Family Sharing.*
- *iCloud Drive.*
- *Apple Music.*
-

iOS 9

Lanzado en septiembre del 2016 junto con el *iPhone 6s* y *iPhone 6s Plus*, para abril del 2016 se lanzó la actualización *iOS 9* para el *iPhone SE*, sus características fueron:

- *Función 3D Touch.*
- Aplicación renovada *Maps* con indicaciones de tránsito.
- *Newsstand.*
- *Wallet.*

Figura 2.4: logos de las versiones *IOS* 7, 8 y 9.
Fuente: [4]

iOS 10

Lanzada en septiembre del 2016 junto al *iPhone 7* y el *iPhone 7 Plus*, la actualización se encuentra disponible para las versiones 5,*5C*,*5S*, *SE*,6,*6 Plus*, *6S* y *6S Plus*.

iOS 11

Lanzada en septiembre del 2017 junto al *iPhone 8*, *iPhone 8 Plus* y *iPhone x*.

Figura 2.5: logos de las versiones *IOS* 10 y 11.
Fuente: [5]

Android

Android es una plataforma completa de código abierto diseñada para dispositivos móviles. Como tal, *Android* está revolucionando el espacio móvil. Por primera vez, se trata de una plataforma que separa el *hardware* del *software* que se ejecuta en él. Esto permite un número mucho mayor de dispositivos para ejecutar las mismas aplicaciones y crea un mucho más rico ecosistema para desarrolladores y consumidores. [6]

Android por ser una plataforma de código abierto, es decir, no se necesita que los desarrolladores y las empresas fabricantes de teléfonos móviles paguen gastos de licencias, para los desarrolladores, no se necesita ningún tipo de licencia para poder comenzar a desarrollar aplicación móvil para esta plataforma, únicamente se

necesita descargar el SDK (*Software Development Kit*) de *Android* para comenzar a desarrollar, el cual es proporcionado en su página oficial.

Este sistema operativo está basado en *Linux* y la idea era tener un nuevo *software* para dispositivos móviles con pantalla táctil como son los teléfonos inteligentes y las *tablets*. El sistema fue desarrollado por *Android Inc*., que *Google* respaldó económicamente en su momento y que más tarde compró en el 2005. *Android* se presentó en el 2007 junto con la *Open Handset Alliance*, un consorcio de compañías de *hardware*, *software* y telecomunicaciones, con la intención de avanzar en los estándares de los sistemas abiertos. El primer teléfono con *Android* fue el HTC(*High Tech Computer Corporation*) *Dream*, que empezó a venderse en septiembre del 2008. [7]

Figura 2.6: *HTC Dream* primer teléfono con *Android*

Fuente: [8]

Características de Android

Actualmente los *Smartphone* son uno de los dispositivos móviles con más demanda en el mercado, el constante avance tecnológico que actualmente vemos en estos dispositivos no solo impulsa a los desarrolladores a buscar nuevas formas de hacer que estos dispositivos sean más delgados y ligeros para el público, actualmente existen diferentes sistemas operativos móviles como son *iOS, Windows Phone,*

Android, Firefox OS, etc. Cada uno de estos sistemas tienen sus características propias que los distinguen de los demás, a continuación, se mencionan las caracterices del sistema operativo *Android*:

- Código abierto: a diferencia de su principal competidor *iOS* que es propiedad de la empresa *Apple*, el cual su sistema solo puede ser montado dentro de sus propios dispositivos, *Android* está disponible para que cualquiera empresa fabricante de teléfonos móviles pueda obtener el sistema y poder adaptarlo a sus dispositivos, esto propicia tener una mayor variedad marcas y modelos de distintos fabricantes a diferencia de los dispositivos que usan *iOS*.
- Núcleo basado en el *kernel* de *Linux*: el cual funciona como el corazón de este sistema operativo, es decir depende de los servicios base que ofrece Linux como son seguridad, gestión de procesos y memoria, una de sus funciones es encargarse de que el *software* y el *hardware* de nuestro dispositivo funcionen y pueda interactuar entre ellos.
- Adaptabilidad a distintos pantallas y resoluciones: actualmente se puede encontrar este sistema en pequeños dispositivos como son los *Smart wach*, la flexibilidad que *Android* ofrece a la adaptación de pantalla brinda a los usuarios una mejor experiencia al momento de usar dispositivos más pequeños que un dispositivo móvil.
- *SQLite*: utilización de *SQLite* para el almacenamiento de datos.
- Soporte de *java* y muchos formatos multimedia.
- *Framework* de aplicaciones: permite el reemplazo y la reutilización de componentes.
- Multitarea real de aplicaciones.
- Optimización del uso de la batería.
- Utilización de sensores: permite el desarrollo de aplicaciones que utilicen los distintos sensores que ofrece el dispositivo para interactuar con el medio que nos rodea.

- *Play store*: ofrece una tienda de aplicaciones donde se puede encontrar una gran variedad de estas.
- Soporte HTML(*HyperText Markup Language*), HTML5(*HyperText Markup Language 5*), XHTML(*Extensible Hypertext Markup Language*), *Adove Flash player*, etc.
- Soporte *BlueTooth*
- Incluye un emulador de dispositivos: herramientas para depuración de memoria y análisis del rendimiento del *software*.

Arquitectura del sistema operativo Android

En la siguiente figura se muestre la pila de *software* que componen la arquitectura de *Android*, las capas que componen la arquitectura esta formadas por *software* libre.

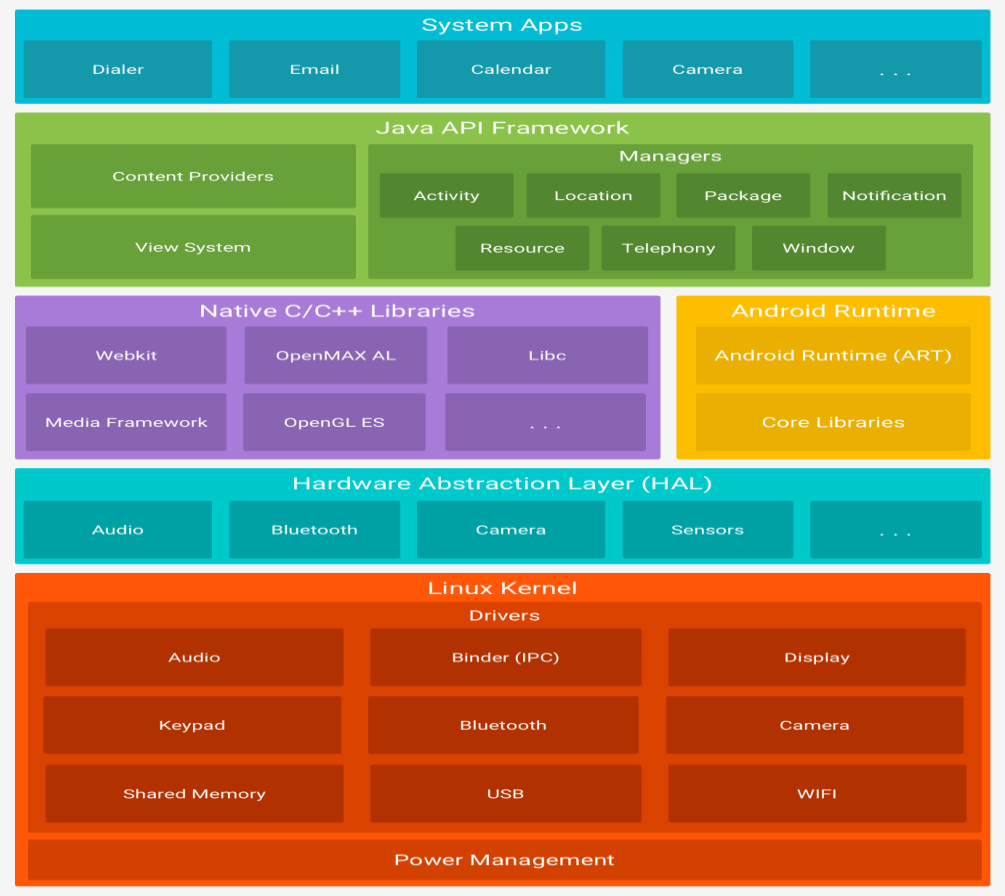


Figura 2.7: arquitectura de *Android*.

Fuente: [9]

Kernel de Linux

Como se ha mencionado anteriormente, la base de esta plataforma es el *Kernel* de *Linux*, su uso permite que Android cuente con los servicios base de este sistema como es la seguridad, la administración de la memoria, la administración de los procesos, este se encarga de comunicar el *software* con el *hardware* de nuestro dispositivo móvil.

Tras la versión *Jelly Bean*, el sistema *Android* está basado en el núcleo de *Linux 3.0*, y las primeras versiones estaban basadas en el núcleo 2.6. Este núcleo tiene en cuenta la gestión de las capas inferiores, tales como los procesos, la gestión de la memoria, los permisos de usuario y la capa de *hardware*. [10]

Hardware Abstraction Layer(HAL)

La capa de abstracción de *hardware* (*HAL*) brinda interfaces estándares que exponen las capacidades de hardware del dispositivo al framework de la eI de nivel más alto. La *HAL(Hardware Abstraction Layer)* consiste en varios módulos de biblioteca y cada uno de estos implementa una interfaz para un tipo específico de componente de *hardware*, como el módulo de la cámara o de bluetooth. Cuando el *framework* de una API (*Application Programming Interface*) realiza una llamada para acceder a hardware del dispositivo, el sistema *Android* carga el módulo de biblioteca para el componente de *hardware* en cuestión. [11]

Android Runtime

Está basado en el concepto de máquina virtual utilizado en *Java*. Dadas las limitaciones de los dispositivos donde ha de correr *Android* (poca memoria y procesador limitado), no fue posible utilizar una máquina virtual *Java* estándar. Google tomó la decisión de crear una nueva, la máquina *virtual Dalvik*, que respondiera mejor a estas limitaciones. Entre las características de la máquina *virtual Dalvik* que facilitan esta optimización de recursos se encuentra la ejecución de ficheros *Dalvik* ejecutables *(.dex)* formato optimizado para ahorrar memoria. Además, está basada en registros. Cada aplicación corre en su propio proceso *Linux* con su propia instancia de la máquina *virtual Dalvik*. Delega al kernel de *Linux*

algunas funciones como *threading* y el manejo de la memoria a bajo nivel. A partir de *Android 5.0* se reemplaza *Dalvik* por *ART*. Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código *Java* hasta en un 33%. También se incluye en el *runtime* de *Android* el módulo Core Libraries, con la mayoría de las librerías disponibles en el lenguaje *Java*. [12]

Native Libraries

Esta capa corresponde a las librerías que *Android* utiliza. Estas librerías se encuentran escritas en *C y C++* utilizadas en varios de los componentes de *Android*. Compiladas para la arquitectura *hardware* especifica del teléfono, tarea que normalmente realiza el fabricante, que también se encarga de instalarlas en el terminal antes de ponerlo a la venta. Su cometido es proporcionar funcionalidad a las aplicaciones, para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma más eficiente. [13]

Entre las librerías más importantes se pueden encontrar las siguientes: [13]

- Librería *libc*: incluye todas las cabeceras y funciones según el estándar del lenguaje *C*. Todas las demás librerías se definen en este lenguaje.
- Librería *Surface Manager*: es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- *OpenGL/SL y SGL*: representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de *Android*. *OpenGL/S*L maneja gráficos en *3D* y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos *3D*. Por otro lado, *SGL*(motor de gráficos *2D*.) proporciona gráficos en *2D*, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en *3D* y *2D*.

- Librería *Media Libraries*: proporciona todos los códecs necesarios para el contenido multimedia soportado en *Android* (vídeo, audio, imágenes estáticas y animadas, etc.)
- *FreeType*: permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.
- Librería SSL (*Secure socket layer*): posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.
- Librería *SQLite*: creación y gestión de bases de datos relacionales.
- Librería *WebKit*: proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma *Android*.

Java API Framework

Esta capa brinda un conjunto de herramientas que son utilizadas por los desarrolladores, brinda las librerías de *java* que se utilizan para la creación de aplicaciones ofreciendo la reutilización de componentes del propio sistema y servicios, los servicios que podemos encontrar son: [14]

- *Activity Manager*: conjunto de API (*Application Programming Interface*) que gestiona el ciclo de vida de las aplicaciones en *Android*.
- *Window Manager*: gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.
- Telephone Manager: incluye todas las API (*Application Programming Interface*) vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- *Content Provider*: permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API (*Application Programming Interface*) la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.
- *View System*: proporciona un gran número de elementos para poder construir interfaces de usuario (*GUI*), como listas, mosaicos, botones, *check-boxes*,

tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.

- *Location Manager*: posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
- *Notification Manager*: mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión *Wi-Fi* disponible, ubicación en un punto determinado, etc. Si llevan asociada alguna acción, en *Android* denominada *Intent*, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple *clic*.

Systems Apps

En la capa superior de la pila de software encontramos todas aquellas aplicaciones del dispositivo, aquellas que cuentan con una interfaz de usuario como las que no la tienen, aplicaciones nativas como administrativas, incluso las que vienen instaladas por el usuario, dentro de esta misma capa encontramos nuestro *launcher* , como sabemos es el encargado de ejecutar otras aplicaciones proporcionando una lista de aplicaciones instaladas, los diferentes escritorios donde se pueden colocar los accesos directos , también podemos encontrar los widgets que son también aplicaciones que vienen en esta capa.

Ventajas y desventajas

Ventajas del sistema *Android:*

- El código de *Android* es abierto: *Google* liberó *Android* bajo licencia *Apache*.
- Variedad: *Android* al ser de código abierto, ofrece una gran variedad de dispositivos móviles con este sistema operativo, ofrece a los consumidores una fácil accesibilidad a esta plataforma a un bajo costo.
- Realidad aumentada.
- Actualizaciones constantes.
- Personalización.

- Acceso a todos los servicios que google ofrece.
- Impulso a los desarrolladores: ofrece a los desarrolladores una gran variedad de herramientas para la creación de aplicaciones móviles.
- *Doze*: sistema administrador de energía que fue lanzado con la versión *Marshmallow*.
- Multiventana: ofrece a los usuarios la posibilidad de usar dos aplicaciones a la vez en pantalla dividida.
- *Play Store*: ofrece una gran variedad de aplicaciones de terceros tanto gratuitas como de paga.

Desventajas del sistema Android

- Código abierto: al ser de código abierto, *Android* es propenso a sufrir ataques.
- Proceso de revisión de *Apps*(*application*): a pesar de ofrecer una gran variedad de aplicaciones, muchas de las aplicaciones que encontramos en *Play store* no cumplen con la calidad esperada, recientemente se han encontrado aplicaciones que se han visto involucradas con el robo de información personal que se encuentran almacenadas en su tienda de aplicaciones.
- Fragmentación del sistema.
- Poco intuitivo: Para la mayoría el sistema operativo es muy complicado.
- Instalación de aplicaciones externas: Muchos de los dispositivos vienen con aplicaciones propias de los operadores de telefonías, esto propicia una gran molestia entre los usuarios.

Versiones de *Android*

Android 1.0 Apple Pie

Esta primera versión fue lanzada el 23 de septiembre del 2008, dicha versión vendría preinstalada en el teléfono *HTC Dream*, sería la primera aparición del

sistema operativo para dispositivos móviles. Entre sus principales características se encuentran:

- Introducción de la tienda de aplicaciones *Android Market*.
- Navegador *web*, capaz de visualizar páginas *web* basadas en HTMLL(*HyperText Markup Language*) y XHTML(*Extensible Hypertext Markup Language*).
- Soporte para la cámara.
- Soporte a los servicios de correo electrónico.
- Sincronización con los servicios de *Google*: *Gmail, Mapas, YouTube.*
- Mensajería SMS(*Short message service*) y MMS(*Multimedia messaging service*)
- Fondos de pantalla y *Widgets.*

Android 1.1 Banana Bread

Esta primera actualización se lanzó el 9 de febrero del 2009, dicha actualización estaba orientada al *HTC Dream* con la finalidad de corregir los errores que se presentaron en la primera versión, entre sus principales cambios se encuentran:

- Actualización para *Google Maps.*
- Archivos adjuntos a mensajes.
- Actualización automática.

Android 1.5 Cupcake.

Esta actualización fue lanzada el 30 de abril del 2009, con el lanzamiento de esta actualización llegaron nuevos cambios en la interfaz de usuario, fue el primer logo con forma de postre, a partir de esta versión de *Android, Google* planteo lanzar las futuras actualizaciones en orden alfabético y con nombre de postres. Sus principales cambios fueron:

- Teclado táctil *qwerty* con predicción de texto.
- Animación en la transición de la pantalla.

- Rotación de pantalla.
- Soporte para la reproducción y grabación de videos.
- Nuevos *widgets* y carpetas.
- Soporte para *BlueTooth*.

Figura 2.8: logos de las versiones *Android*, *1.0 Apple pie,1.1 Banana Bread, 1.5 Cupcake.*
Fuente: [15]

Android 1.6 Donut

Lanzada el 15 de septiembre del 2009, lanzada como una actualización para los nuevos dispositivos, sus principales cambios fueron:

- Cuadros de búsqueda.
- Adaptabilidad a toda resolución de pantalla.
- Rediseño para *Android Market*.
- Integración de nueva interfaz para la cámara
- Integración de la galería y nuevas funcionalidades.
- Actualización para la búsqueda por voz.

Android 2.0 / 2.1 Eclair

Lanzada el 26 de octubre del 2009, sus principales cambios fueron:

- Navegación de GPS (sistema de posicionamiento global) de *Google Maps*.
- Personalización de la pantalla de inicio con fondos de pantalla animados.
- Texto a voz.

- Soporte para HTML5(*HyperText Markup Language* 5).
- Soporte para flash, *zoom* y balanceo de blanco.
- Velocidad de *hardware* optimizada.

Android 2.2 Froyo

Lanzada el 20 de mayo del 2010, el *Nexus One* fue el primer dispositivo actualizado a la versión 2.2. Sus cambios fueron:

- Mejoras en cuanto al rendimiento.
- Desbloqueo *Pin*(Número de identificación personal).
- Función *wifi Hostpod*.
- Actualizaciones a través de anudrid Market.
- opción de mover aplicaciones a la tarjeta *SD*(*Secure digital*).

Figura 2.9: logos de las versiones *Android*, *1.6 Donut,2.0/2.1 Eclair, 2.2 Froyo*
Fuente: [16] [17] [18]

Android 2.3 GingerBread

Lanzada el 6 de diciembre del 2010, esta versión vendría con el Nexus S sus cambios incluyeron:

- API(*Application Programming Interface*) de videojuegos.
- NFC*(Near field communication).*
- Administración de la batería.

- Mejora en el teclado.
- Soporte para la cámara frontal.
- Mejora en el control de copiar y pegar.
- Soporte para pantallas más grandes.
- Soportes de audio.
- Soporte nativo para sensores.

Android 3.0 / 3.1 Honeycomb

Lanzado el 22 de febrero del 201, esta versión solo estuvo disponible para *tablets*, la tableta *Xoom* perteneciente a Motorola fue la primera en salir con esta versión. Sus cambios fueron:

- Renovada interfaz de usuario.
- Escritorio *3D widgets* rediseñados
- Sistema multitarea mejorado
- Soporte para video *chat*
- Soporte para accesorio *USB (Universal serial bus)*
- Final de los botones físicos

Android 4.0 Ice Cream Sandwich

Lanzada el 19 de octubre del 2011, *Ice Cream Sándwich* subió la apuesta en cuanto a la personalización y el control por parte del usuario. Permitió personalizar la pantalla principal, definir el uso de datos y compartir contenido al instante [11], sus cambios fueron:

- Pantalla principal personalizada.
- Control de uso de datos.
- *Android Beam.*
- Botones virtuales.
- Sistema unificado para *Smartphone* y tabletas.
- Multitarea mejorada.

- Corrector de texto rediseñado.
- Captura de pantalla al presionar la tecla bajar volumen y botón de encendido.
- Reconocimiento de voz de usuario.
- Actualización para la cámara, introducción de fotografía panorámica.
- Actualización para la cámara, introducción de fotografía panorámica.

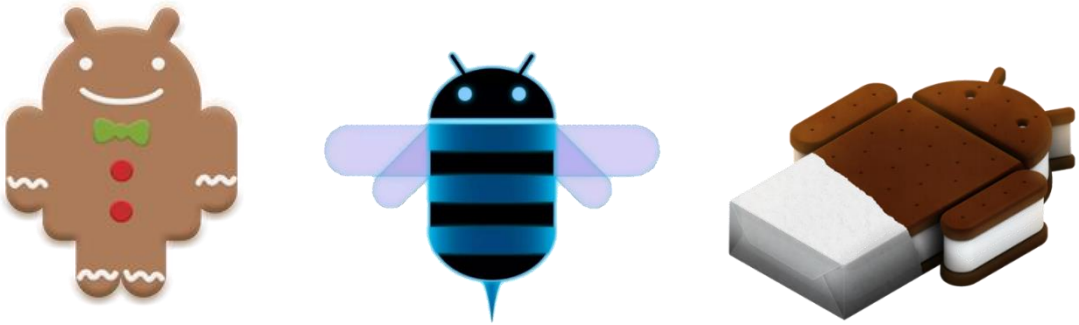


Figura 2.10: logos de las versiones *Android*, *2.3 GingerBread, 3.0/3.1 Honeycomb, 4.0 Ice Cream Sandwich*
Fuente: [19] [20]

Android 4.1/ 4.2 /4.3 Jelly Bean

Lanzado el 27 de junio del 2012 y lanzado al mercado el 13 de julio junto con el *Nexus 7*, la inteligencia fue una de las facetas principales de *Jelly Bean*, que marcó el comienzo de la era de la asistencia móvil personalizada gracias a *Google Now*. De esta manera, las notificaciones se volvieron más accionables y fue posible que un dispositivo funcionara con varias cuentas de usuario [11], sus cambios fueron:

- Mejora en la fluidez y la estabilidad.
- Ajuste de widgets al ser añadidos al escritorio.
- Mejora en el dictado de voz con la posibilidad de utilizarlo sin *internet*.
- Nuevas lenguas adicionadas
- Transmisión de videos a través de NFC *(Near field communication).*
- Notificaciones mejoradas.
- *Google Chrome* se vuelve el navegador predeterminado.
- Pequeños cambios en la interfaz de usuario.

Android 4.4 KitKat

Lanzada el 31 de octubre del 2013, Android KitKat permitió realizar tareas con el sonido de tu voz, solo se tenía que decir “Ok Google” para iniciar búsquedas por voz, enviar un mensaje de texto, obtener indicaciones sobre cómo llegar a un lugar o reproducir una canción. [11] Sus cambios fueron:

- Voz *Ok Google.*
- Nuevo diseño.
- Marcador inteligente.
- *Google Drive Box.*
- Mejora en la multitarea.
- Optimización del rendimiento de la memoria.

Android 5.0 Lollipop

Lanzado el 12 de noviembre del 2014, *Lollipop* es hasta el momento, siendo octubre del 2017, la versión de Android más extendida a nivel mundial y hoy en día, la gran mayoría de dispositivos *Android* cuentan con esta versión de sistema operativo. El tema de mantener nuestra información sincronizada en diversos dispositivos es uno de los fines que cumple *Lollipop* [21], sus cambios fueron:

- *Material Desing.*
- Nuevo teclado.
- Nueva forma de controlar el consumo de batería.
- Pantallas múltiples.
- Notificaciones.
- Integración con *smartwatches.*
- Soporte para procesadores de 64 bits.
- Desbloqueo por ubicación.
- Cambio visual en las multitareas acoplando las ventanas en tarjetas.

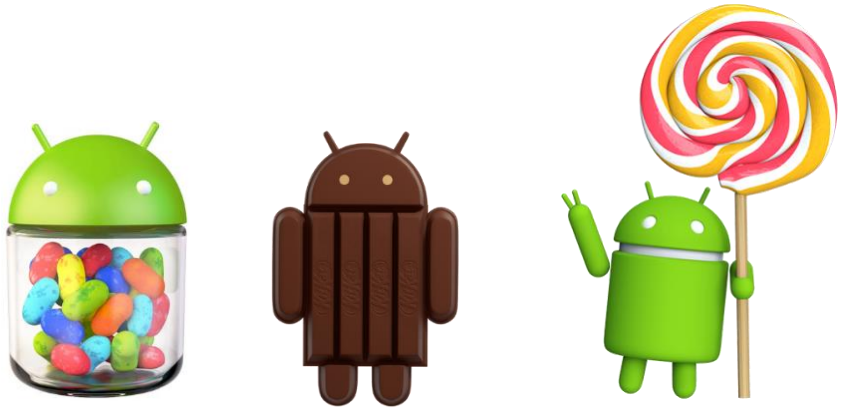


Figura 2.11: logos de las versiones *Android*, *4.1/4.2/4.3 Jelly Bean, 4.4 Kitkat, 5.0 Lollipop*
Fuente: [22] [20] [23]

Android 6.0 Marshmallow

Lanzada el 5 de octubre del 2015, Luego de meses de espera, finalmente *Google* lanzó *Android 6.0* junto a los nuevos dispositivos *Nexus de LG* y *Huawei*, sus cambios fueron: [21].

- Google Now on Tap.
- Permisos de aplicaciones caso por caso.
- Mejoras en la gestión de la batería gracias a Doze.
- Mejoras en las funciones copiar, cortar y pegar.
- Soporte para huellas dactilares.
- Nuevo USB(Universal serial bus) Tipo-C y Chrome funcionando dentro de otras aplicaciones.

Android 7.0 Nougat

Presentado durante el evento Google I/O en mayo del 2016, *Nougat* se resume como una actualización de las novedades antes mencionadas en *Marshmallow*, la anterior versión de *Android*, sus cambios fueron: [21]

- Multiventana
- *Android Nougat* soporta de forma nativa la realidad virtual.
- Lanzamiento e introducción de una tienda de aplicaciones tipo Google Play pero dedicada a la realidad virutal: *DayDream*

- *Doze*, el sistema de administración de energía que fue presentado en Marshmallow, ahora ha sido mejorado con *Doze on the Go.*
- Configuración regional de varios idiomas.
- La *API* (*Application Programming Interface*) de *vulkan*

Android 8.0 Oreo

El nombre en código de la siguiente versión de Android es *Android Oreo* y fue anunciado por *Google* el pasado 21 de marzo del 2017 siendo que su primera versión “Alfa” fue publicada para dispositivos *Google Pixel* y *Nexus* el 22 de Marzo del 2017, presenta las siguientes novedades: [21].

- Optimización de procesos en segundo plano para optimizar aún más el consumo de batería.
- Ahora las notificaciones se podrán organizar dependiendo de su categoría y se podrá decidir la frecuencia con la que las recibimos en nuestros Smartphone, además al deslizar la notificación a la izquierda, aparecerá un recuadro de ajustes para controlar las notificaciones de cada aplicación, así como un reloj para definir la periodicidad con la que recibimos dichas notificaciones
- Se han realizado cambios en el diseño del apartado de ajustes, logrando un aspecto más claro y menos pesado
- Google ha actualizado la guía para diseñar iconos, de modo que se puedan unificar los diseños, ya que ahora habrá iconos animados en dos capaz que podrán mostrar información sin necesidad de abrir las aplicaciones
- Como ya está funcionando en algunas Apps como Snapchat o Facebook, aparecerán números al lado de los íconos de algunas aplicaciones para denotar que hay cosas pendientes de revisar.

Figura 2.12: logos de las versiones *Android*, *6.0 Marshmallow,* 7.0 Nougat *, 8.0 Oreo*

Fuente: [24] [25] [26]

Estadística de Usuarios que utilizan el sistema operativo *Android*

Tomando como referencia los servicios de estadística que ofrece *NetMarketShare*, la cuota de mercado de sistemas operativos móviles a inicios del 2017 es el siguiente: [27].

- Android 66,71 %.
- iOS 29,55 %.
- Windows Phone 1,41 %.
- BlackBerry OS 0,37 %.

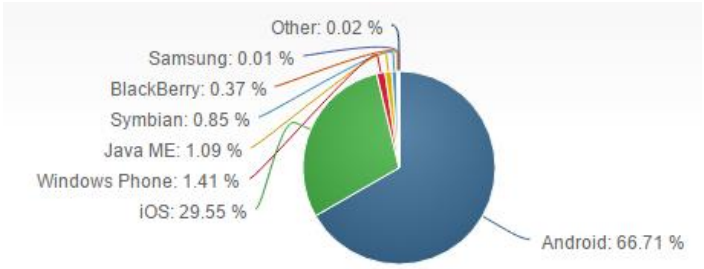


Figura 2.13: estadística de sistemas operativos móviles a inicios del 2017

Fuente: [27]

A pesar de que *Android* ocupa el 66.71% del mercado en el 2017, no todos los usuarios han podido acceder a las versiones más actuales de este sistema, actualmente ya vamos en la versión *Oreo 8.0* pero a pesar de eso menos del 2% de los dispositivos móviles cuentan con esta actualización, *Lollipop* es hasta ahora la versión más utilizada hasta ahora.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.0%
4.2.x		17	3.0%
4.3		18	0.9%
4.4	KitKat	19	13.4%
5.0	Lollipop	21	6.1%
5.1		22	20.2%
6.0	Marshmallow	23	29.7%
7.0	Nougat	24	19.3%
7.1		25	4.0%
8.0	Oreo	26	0.5%

Datos recopilados durante un período de 7 días hasta 11/12/2017.
No se muestran versiones con una distribución inferior al 0,1%.

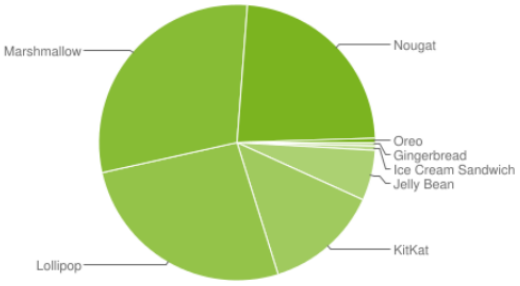


Figura 2.14: porcentaje de versiones *Android* distribuidas

Fuente: [28]

2.2. Entornos para desarrollo de Apps en Android

Para iniciar con el desarrollo de nuestras propias aplicaciones móviles es necesario que contemos con un entorno de desarrollo que nos ofrezca las herramientas necesarias, una interfaz que nos ayude a la manipulación de elementos de diseño como en este caso son el uso de botones, etiquetas, cajas de texto, *layouts* y uso de listas por mencionar algunos de los controles más utilizados, debe proporcionar distintas opciones de dispositivos móviles físicos y virtuales para poder correr nuestras aplicaciones.

Existen muchos *IDE's*(Entorno de desarrollo integrado) para el desarrollo de aplicaciones, entre los más utilizados podemos encontrar: *Android Studio, Eclipse, Netbeans, Intellij y AIDE* por mencionar algunos, *Google* nos ofrece un *IDE*(Entorno de desarrollo integrado) oficial para el desarrollo de aplicaciones, *Android Studio*, antes del lanzamiento de *Android Studio*, *Eclipse* fue uno de los entornos de desarrollo más utilizado por los programadores, la fácil instalación de sus extensiones y librerías generaba un entorno óptimo para los programadores.

A continuación, nos enfocaremos en *Eclipse* y *Android Studio*, dos de los *IDE´s*(Entorno de desarrollo integrado) más utilizados por los desarrolladores de aplicaciones móviles.

2.2.1. Eclipse

Eclipse es un entorno de desarrollo integrado de código abiar creado originalmente por IBM(*International business machines*) aunque actualmente la fundación *Eclipse* en encarga de su desarrollo, *Eclipse* es una plataforma de desarrollo que está diseñada para extenderse a través de plug-ins, a pesar de ser uno de los entornos de desarrollo más utilizado por los desarrolladores de aplicaciones del lenguaje java este no se está pensado para trabajar con un lenguaje en específico, puede ser utilizado para el desarrollo de aplicaciones en *C, C++, php*, etc. Alguna de sus características son las siguientes:

- Código abierto.
- Extensa colección de *Plug-ins.*
- Depurador de código.
- Pruebas unitarias en *Junit.*
- Integración con *Ant*, asistentes para creacion de proyectos,clases,etc.
- Compilación en tiempo real.
- Editor de texto con resaltado de sintaxis.

Figura 2.15: logo de *Eclipse*

Fuente: [29]

2.2.2. Android Studio

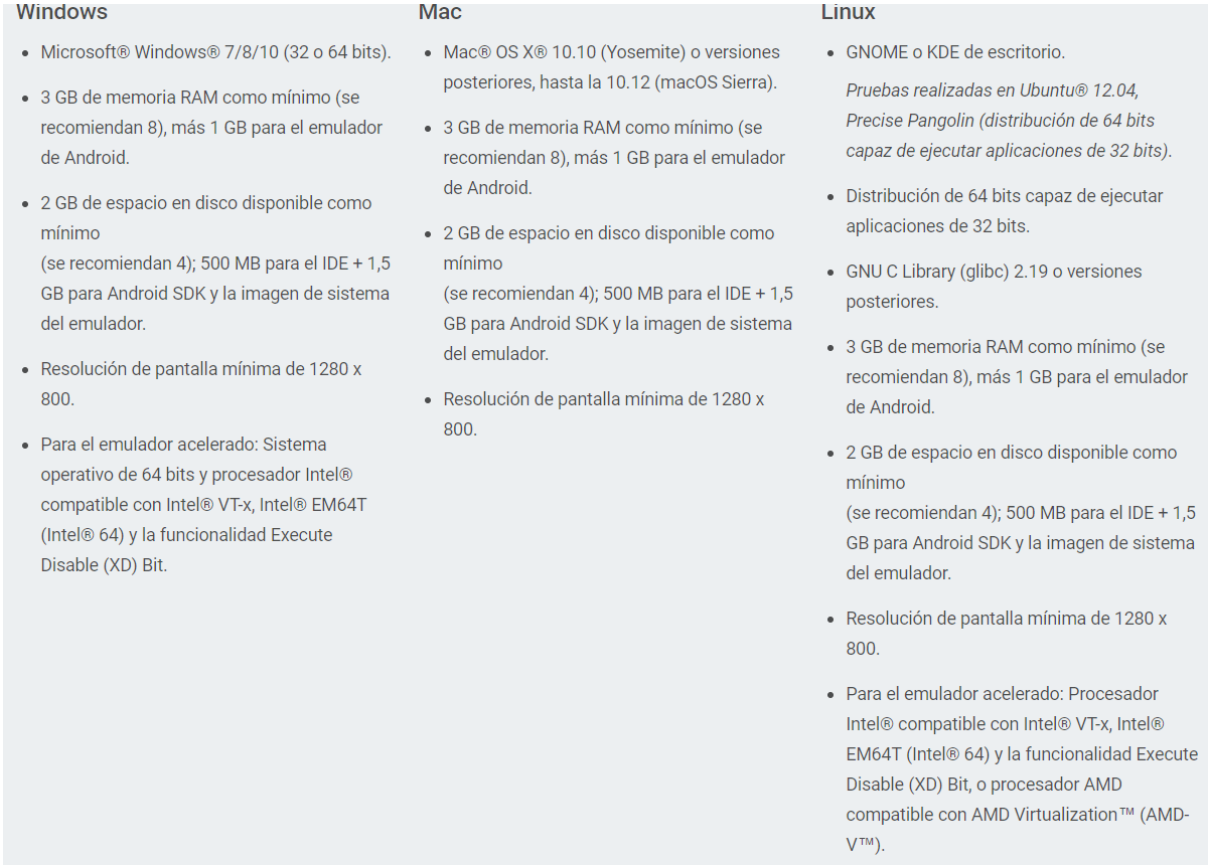
Android Studio es el *IDE* (Entorno de desarrollo integrado) oficial que Google ofrece a los desarrolladores para la creación de aplicaciones móviles Android, Android Studio lanzado el 16 de mayo del 2012 en la Google I/O, para diciembre del 2014 lanza su primera versión estable, fue creado con la intención de sustituir a Eclipse como principal entorno de desarrollo de aplicaciones. Algunas de sus características son las siguientes:

- Soporte para programar aplicaciones para *Android Wear* (sistema operativo para dispositivos corporales como por ejemplo un reloj).
- Herramientas *Lint* (detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador) para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.
- Nuevo diseño del editor con soporte para la edición de temas.
- Nueva interfaz específica para el desarrollo en *Android*.
- Permite la importación de proyectos realizados en el entorno Eclipse, que a diferencia de *Android Studio* (*Gradle*) utiliza *ANT*.
- Posibilita el control de versiones accediendo a un repositorio desde el que poder descargar *Mercurial*, *Git*, *Github* o Subversion.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.
- Integración con *Google Cloud Platform*, para el acceso a los diferentes servicios que proporciona *Google* en la nube.

- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo *xml*.
- Un sistema de compilación basado en *Gradle* flexible.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos *Android*.
- *Instant Run* para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK (*Android aplication package*).
- Gran cantidad de herramientas y *frameworks* de prueba.

Requisitos

Android Studio se encuentra disponibles para *Windows, Mac y Linux*, a pesar de ofrecer una gran variedad de herramientas para el desarrollo se necesita cumplir una serie de requisitos que se necesitan cumplir necesariamente para poder ejecutar este programa de manera eficiente en nuestras computadoras, los requisitos se muestran a continuación:



Windows

- Microsoft® Windows® 7/8/10 (32 o 64 bits).
- 3 GB de memoria RAM como mínimo (se recomiendan 8), más 1 GB para el emulador de Android.
- 2 GB de espacio en disco disponible como mínimo
  (se recomiendan 4); 500 MB para el IDE + 1,5 GB para Android SDK y la imagen de sistema del emulador.
- Resolución de pantalla mínima de 1280 x 800.
- Para el emulador acelerado: Sistema operativo de 64 bits y procesador Intel® compatible con Intel® VT-x, Intel® EM64T (Intel® 64) y la funcionalidad Execute Disable (XD) Bit.

Mac

- Mac® OS X® 10.10 (Yosemite) o versiones posteriores, hasta la 10.12 (macOS Sierra).
- 3 GB de memoria RAM como mínimo (se recomiendan 8), más 1 GB para el emulador de Android.
- 2 GB de espacio en disco disponible como mínimo
  (se recomiendan 4); 500 MB para el IDE + 1,5 GB para Android SDK y la imagen de sistema del emulador.
- Resolución de pantalla mínima de 1280 x 800.

Linux

- GNOME o KDE de escritorio.
  Pruebas realizadas en Ubuntu® 12.04, Precise Pangolin (distribución de 64 bits capaz de ejecutar aplicaciones de 32 bits).
- Distribución de 64 bits capaz de ejecutar aplicaciones de 32 bits.
- GNU C Library (glibc) 2.19 o versiones posteriores.
- 3 GB de memoria RAM como mínimo (se recomiendan 8), más 1 GB para el emulador de Android.
- 2 GB de espacio en disco disponible como mínimo
  (se recomiendan 4); 500 MB para el IDE + 1,5 GB para Android SDK y la imagen de sistema del emulador.
- Resolución de pantalla mínima de 1280 x 800.
- Para el emulador acelerado: Procesador Intel® compatible con Intel® VT-x, Intel® EM64T (Intel® 64) y la funcionalidad Execute Disable (XD) Bit, o procesador AMD compatible con AMD Virtualization™ (AMD-V™).

Figura 2.16: requisitos de *Android Studio*

Fuente: [30]

Instalación de Android Studio en Windows

Paso 1.

Para comenzar con la instalación se es necesario ingresar a la página oficial de Android, para ingresar basta con ingresar este link dentro de nuestro navegador: https://developer.android.com/studio/index.html. Si eres usuario de Windows basta con solo dar un clic en el botón verde que dice descargar Android Studio 3.0.1, véase en la figura 2.17.

Figura 2.17: página para descargar *Android Studio*

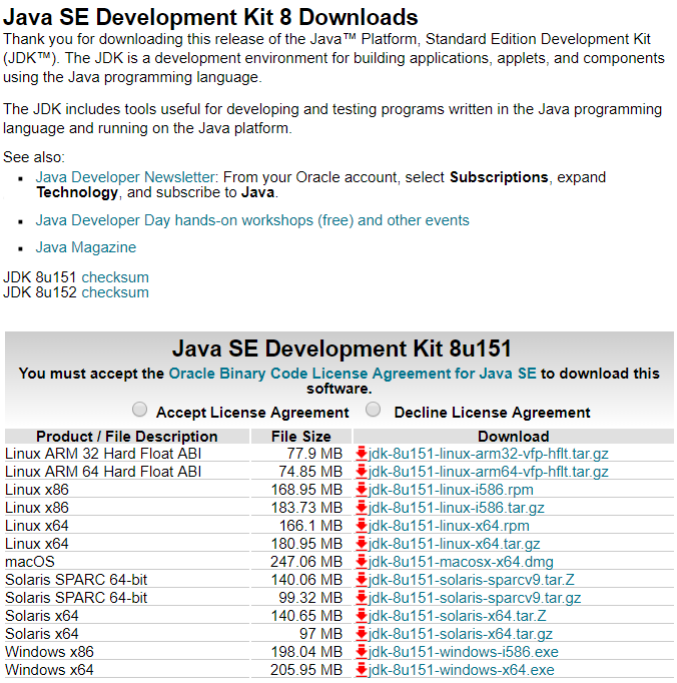
Fuente: [30]

Paso 2.

Procederemos a instalar el JDK (*Java Development Kit*) de java, para ello basta con ingresar en la siguiente dirección para descargarlo:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

aceptamos la licencia y procedemos a descargarlo para la plataforma que estemos ocupando, en mi caso será Windows x64, véase en la figura 2.18.



Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 8u151 checksum
JDK 8u152 checksum

Java SE Development Kit 8u151

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.9 MB	jdk-8u151-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.85 MB	jdk-8u151-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.95 MB	jdk-8u151-linux-i586.rpm
Linux x86	183.73 MB	jdk-8u151-linux-i586.tar.gz
Linux x64	166.1 MB	jdk-8u151-linux-x64.rpm
Linux x64	180.95 MB	jdk-8u151-linux-x64.tar.gz
macOS	247.06 MB	jdk-8u151-macosx-x64.dmg
Solaris SPARC 64-bit	140.06 MB	jdk-8u151-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.32 MB	jdk-8u151-solaris-sparcv9.tar.gz
Solaris x64	140.65 MB	jdk-8u151-solaris-x64.tar.Z
Solaris x64	97 MB	jdk-8u151-solaris-x64.tar.gz
Windows x86	198.04 MB	jdk-8u151-windows-i586.exe
Windows x64	205.95 MB	jdk-8u151-windows-x64.exe

Figura 2.18: página para descargar el jdk de java

Fuente: [31]

Paso 3.

Una vez instalado el JDK(*Java Development Kit*) procederemos a instalar Android Studio, para ello nos ubicaremos en nuestra carpeta descargas o en donde se almacenará el instalador que descargamos de la página de *Android*.

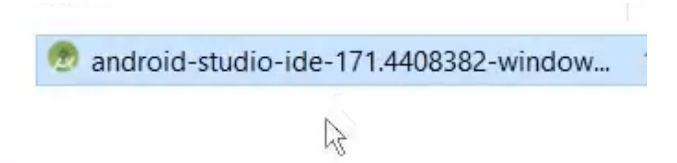


Figura 2.19: archivo instalador de *Android Studio*

Fuente: edición propia

A continuación, nos aparecerá el instalador como se muestra en la siguiente figura, procederemos a dar en *next* para comenzar la instalación.

Figura 2.20: asistente de instalación *Android Studio*

Fuente: edición propia

Seleccionamos los componentes a instalar y damos *Next*.

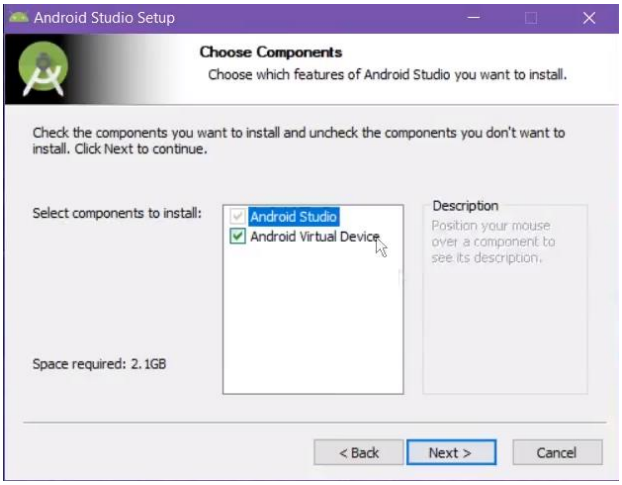


Figura 2.21: selección de componentes

Fuente: edición propia

Seleccionamos la ruta donde se encontrará ubicado *Android Studio*, por recomendación se debe dejar por defecto la dirección que nos aparece en la siguiente figura, en caso contrario solo se debe poner la nueva ubicación.

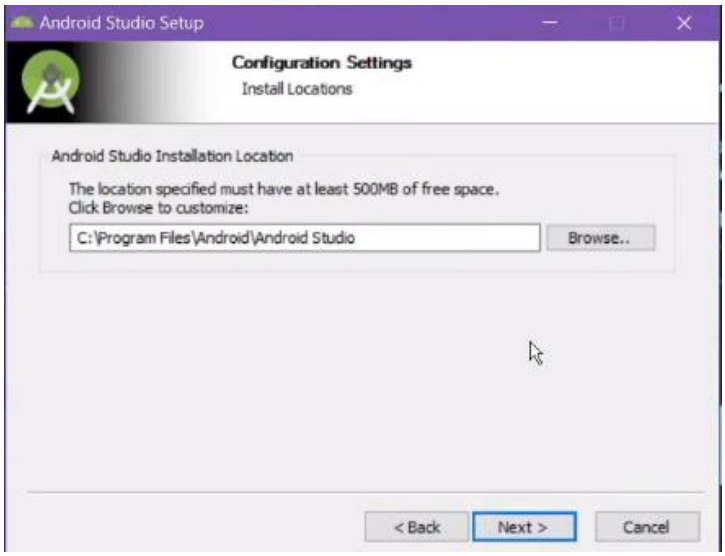


Figura 2.22: ruta de instalación

Fuente: edición propia

Esperamos a que termine de instalarse

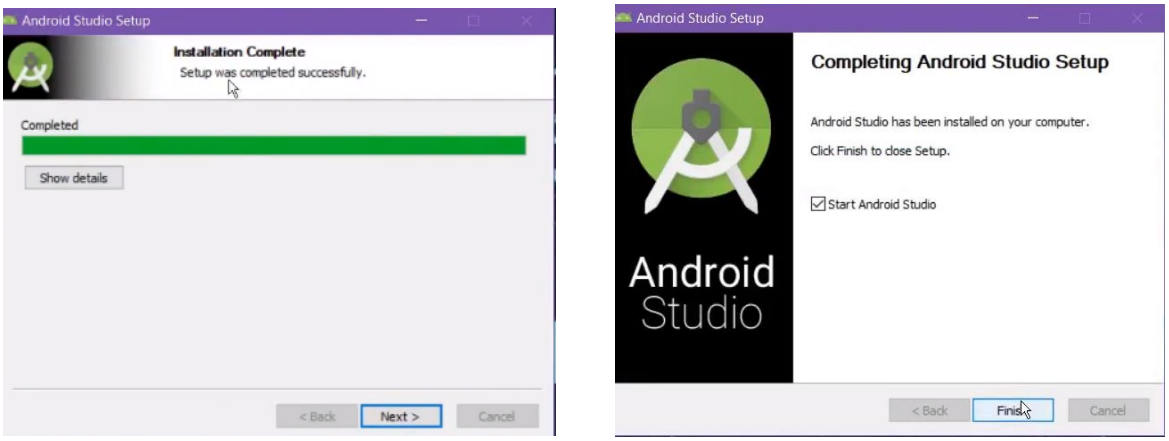


Figura 2.23: finalización de la instalación

Fuente: edición propia

Una vez finalizado nos mostrara un cuadro de opciones como se ve en la siguiente figura, en caso de tener una versión de *Android Studio* anteriormente y deseas importar tus configuraciones seleccionaremos *Import Studio settings from*, en nuestro caso no contamos con ninguna y seleccionaremos la opción *Do not import settings*.

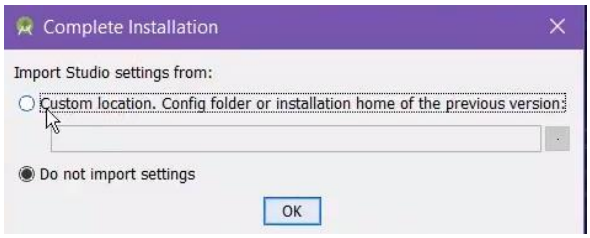


Figura 2.24: importar configuraciones

Fuente: edición propia

Una vez finalizado, nos mostrara una imagen como en la figura 2.25.

Figura 2.25: cargando componentes

Fuente: edición propia

A continuación, se iniciará el asistente de instalación como se muestra en la siguiente figura. Dar clic en *Next* para proseguir con la instalación.

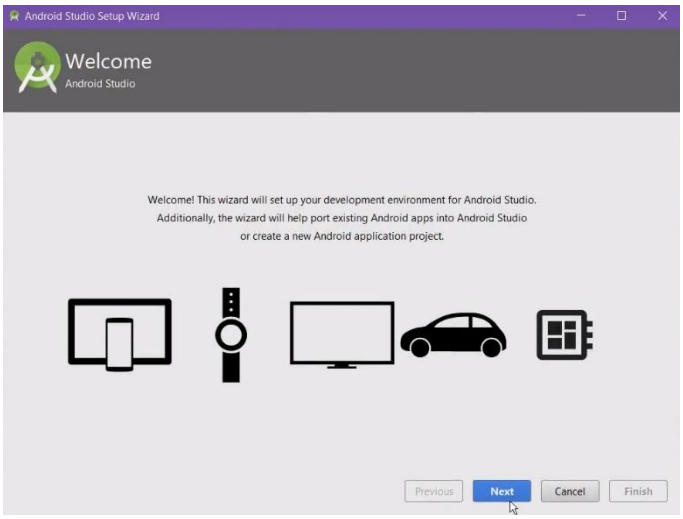


Figura 2.26: ventana de bienvenida

Fuente: edición propia

En la figura 2.27 se muestran las opciones del tipo de instalación que se desea realizar, en este caso selecciona la opción estándar y proseguir con la instalación dando *Next*.

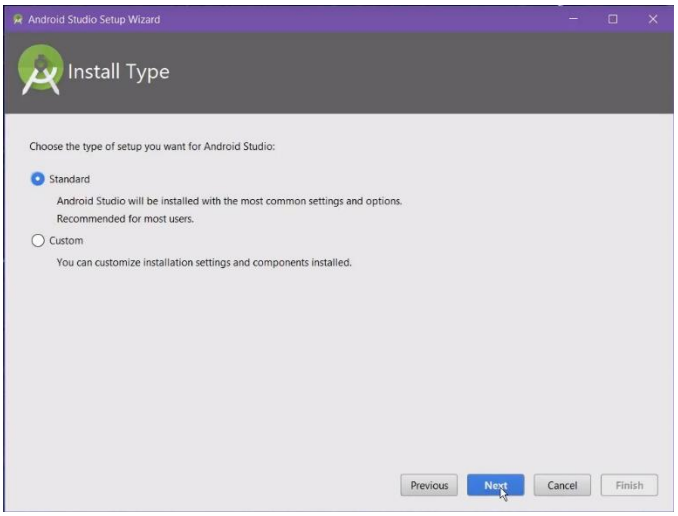


Figura 2.27: tipo de instalación

Fuente: edición propia

En la figura 2.18 se muestra el tipo de tema. Selecciona el que más guste y da clic en *Next.*

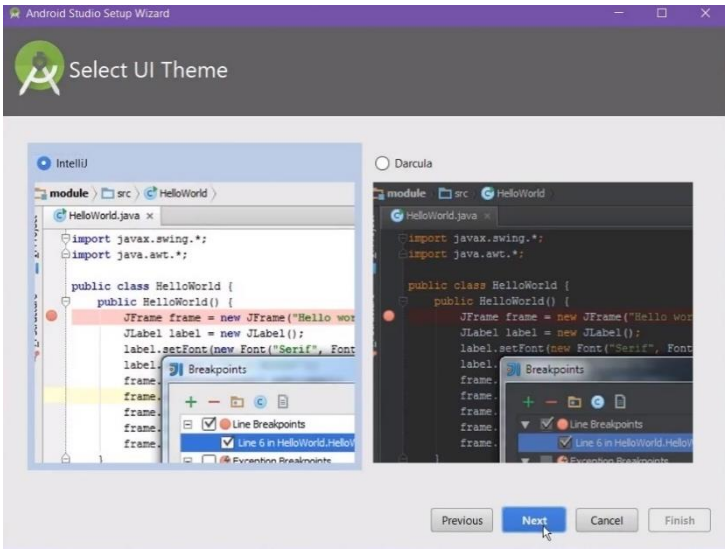


Figura 2.28: selección de tema

Fuente: edición propia

En la figura 2.29 se muestra los componentes que se desean instalar o actualizar, es importante que marque la opción *Android Virtual divice* ya que si no es marcadas no podrá ocupar su propio teléfono para correr aplicaciones, esta opción es muy importante en caso de no tener una computadora con grandes recursos. Dar clic en *Next*.

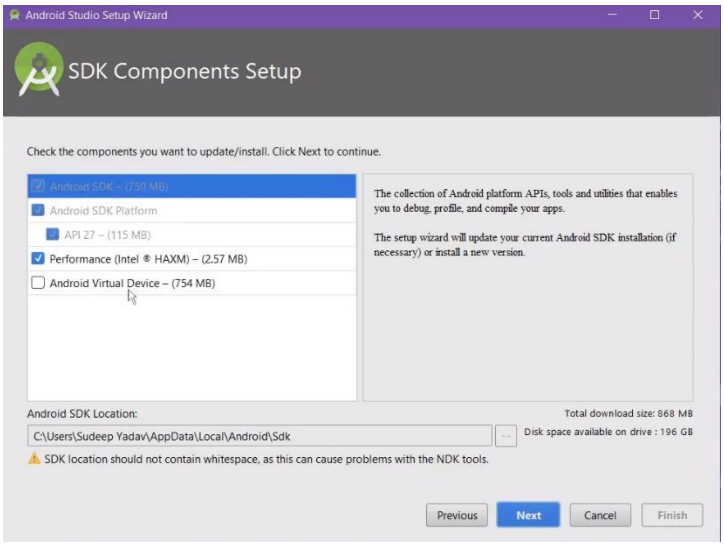


Figura 2.29: componentes de configuración del *SDK*

Fuente: edición propia

En la figura 2.30 se muestra una lista de las configuraciones que acaba de realizar, en caso de no tener ningún inconveniente dar *clic* en *Finish*. Se mostrará el progreso de la descarga de los componentes.

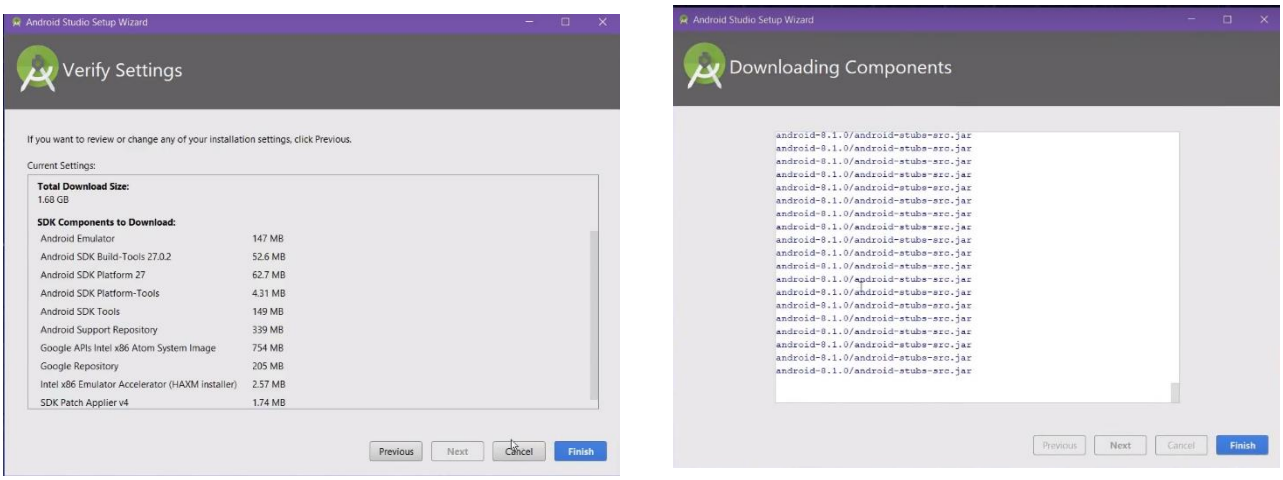


Figura 2.30: finalización de la instalación

Fuente: edición propia

Configuración y creación de un proyecto en Android Studio

Una vez finalizada la instalación de los componentes se mostrará una pantalla como se muestra en la figura 2.31. Para descargar las versiones de Android posicionarse en la parte inferior derecha en *configure* después seleccionaremos *SDK Manager.*

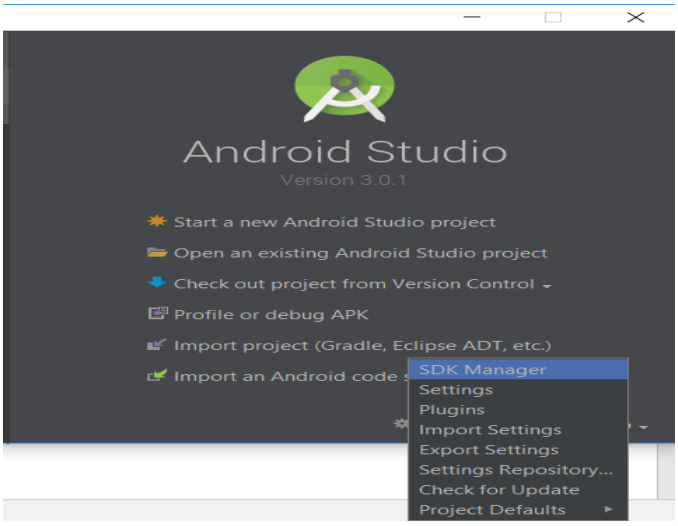


Figura 2.31: menú de opciones

Fuente: edición propia

En la figura 2.32 se muestran aquellos componentes que marcaran, las versiones de Android que se deseen descargar y dar instalar. En este caso se tiene instalado *Jelly Bean, Lollipop y Marshmallow.*

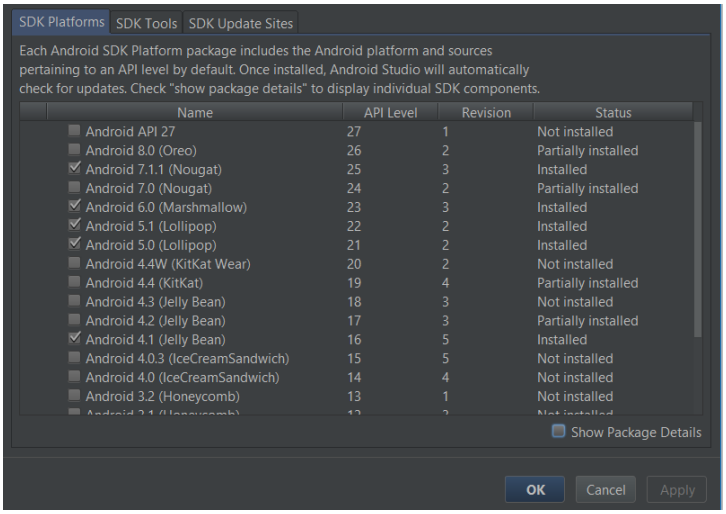


Figura 2.32: ventana del *SDK Manager*

Fuente: edición propia

Ya finalizada la instalación dar ok para volver a la ventana principal, en esta ventana seleccionar la primera opción *Start a new Android proyect.*

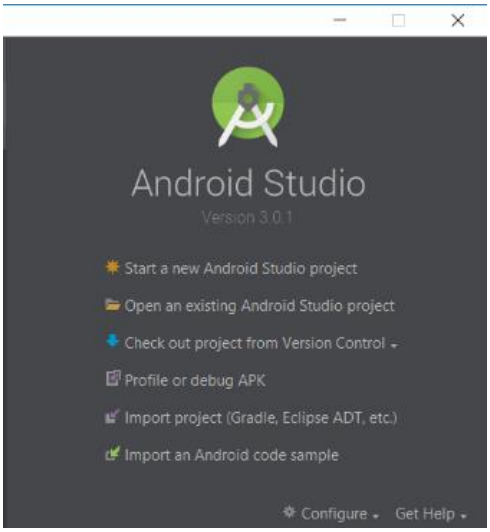


Figura 2.33: ventana principal

Fuente: edición propia

En la figura 2.34 se muestra la ventana se asigna nombre del proyecto, como se observa esta versión de Android Studio ya trae soporte para el lenguaje *Kotlin, Kotlin*

es el nuevo lenguaje de programación que *Google* está introduciendo para programación de aplicaciones Android. Pondrá como nombre al nuevo proyecto *Primer Proyecto* y dejar el resto de la configuración como aparece por defecto.

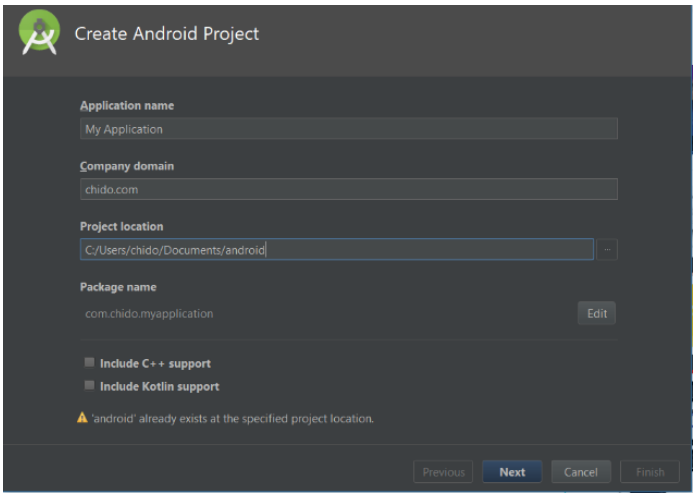


Figura 2.34: creación de un proyecto

Fuente: edición propia

En la figura 2.35 se muestra la ventana en la cual pondrá seleccionar para que tipo de dispositivo va orientada la aplicación, en esta ocasión solo marcar la primera opción que se muestra *Phone and Tablet*, si desplegas la pestaña de esta opción podrá observar que se puede escoger para la versión mínima que desea que la aplicación pueda ser ejecutada. Seleccionar la versión *5.0 Lollipop*.

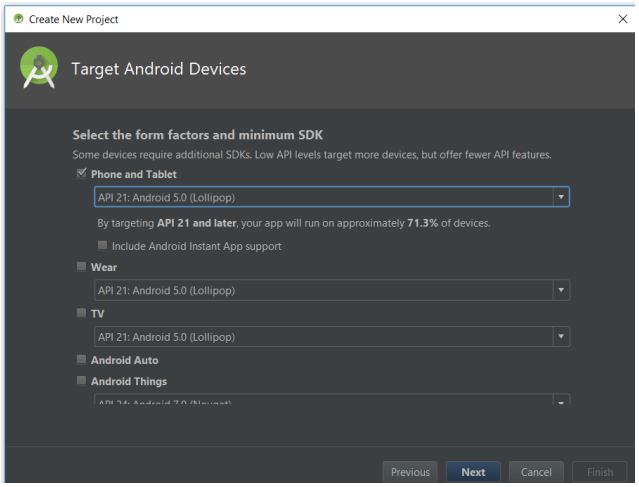


Figura 2.35: elección del tipo de dispositivo

Fuente: edición propia

La figura 2.36 muestra la ventana donde seleccionar el tipo de actividad que se desea crear como actividad principal, dentro de esta ventana se encontraran todo tipo de actividad como pueden ser *Login*, mapas o en blanco, en este caso seleccionar la actividad vacía, seleccionar *Empty Activity.*

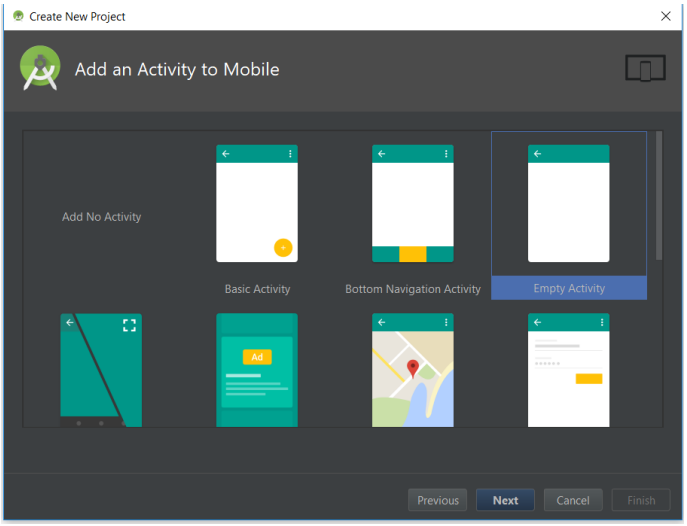


Figura 2.36: agregar una actividad

Fuente: edición propia

La figura 2.37 muestra la ventana donde realizar la configuración que desea hacer a la actividad, para este caso se usara por defecto. Finalmente dar clic en *finish* para que el proyecto sea creado.

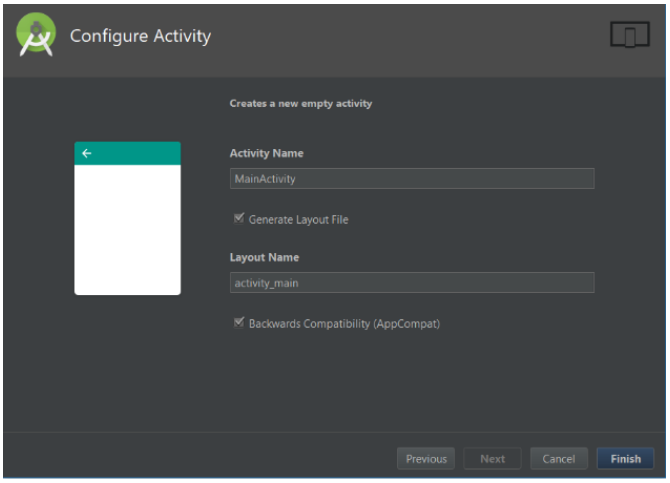


Figura 2.37: Configuración de actividad

Fuente: edición propia

Una vez que se finalizó de crear el proyecto tendra una pantalla como se muestra en la figura 2.38.

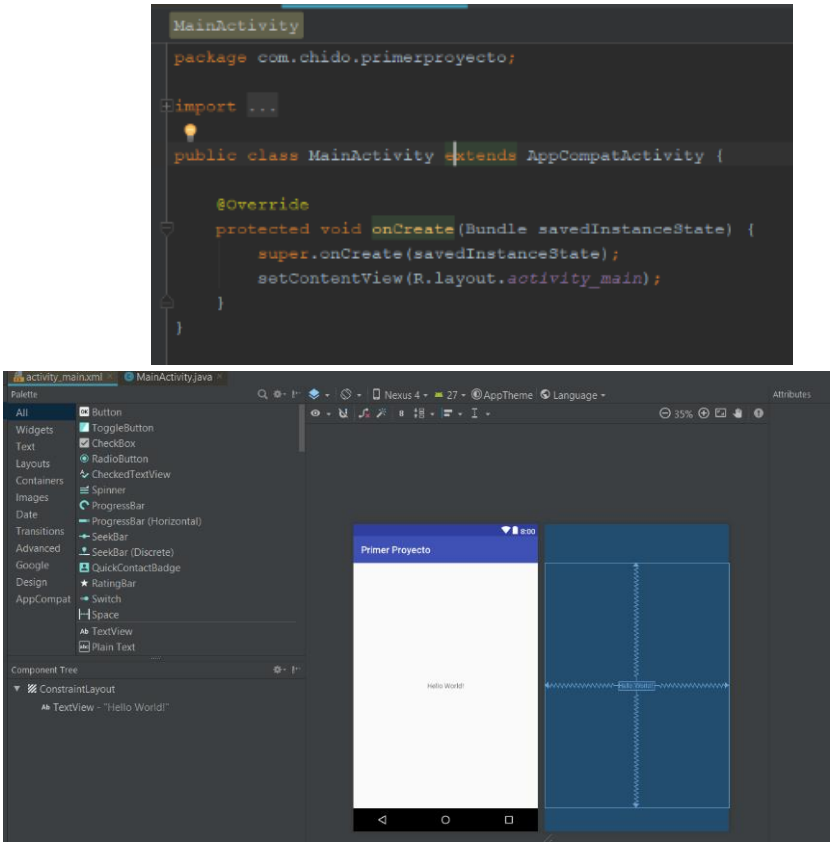


Figura 2.38: ventanas de *Android Studio*

Fuente: edición propia

Configuración de un emulador.

Para poder testear las aplicaciones que se van desarrollando es necesario la creación de un emulador, *Android Studio* ofrece una herramienta muy útil llamada *AVD Manager* (Administrador de Dispositivos Virtuales), en caso de no contar con un Smartphone físico o no contar con una versión más actualizada de Android, esta herramienta te será de mucha ayuda ya que proporciona una gran variedad de dispositivos de los cuales podrás crear tu emulador. Para entrar al *AVD Manager* será necesario buscar un icono como se muestra en la figura 2.39.

.

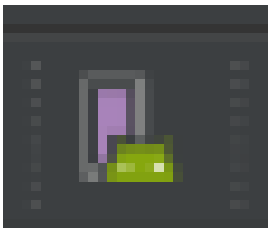


Figura 2.39: logo AVD Manager

Fuente: edición propia

A continuación, en la figura 2.40 se muestra la ventana en la cual podrá visualizar todas aquellas máquinas virtuales que se han creado, en caso de ser la primera vez que instala *Android Studio* no se mostrará ninguna, para proceder a la creación de una, basta con dirigirse al botón que se encuentra en la parte inferior izquierda el cual contiene la etiqueta *créate a new virtual device*. Ya realizadas las instrucciones anteriores, se muestra una nueva venta adonde podrá ver una gran variedad de dispositivos que tiene a su servicio, se selecciona el que se desea utilizar, ya seleccionado se procede a dar clic sobre el botón con la etiqueta *Hardware Profile*.

Dentro de esta nueva ventana podrá ver y asignar las configuraciones de *harware* a nuestra máquina virtual.

En esta ventana podrá configurar el dispositivo, memora *Ram* que se le asignara, servicios como sensores, GPS (sistema de posicionamiento global)., cámara y giroscopio, dejaremos todo por default y le asignaremos el nombre Primer AVD(Administrador de Dispositivos Virtuales).

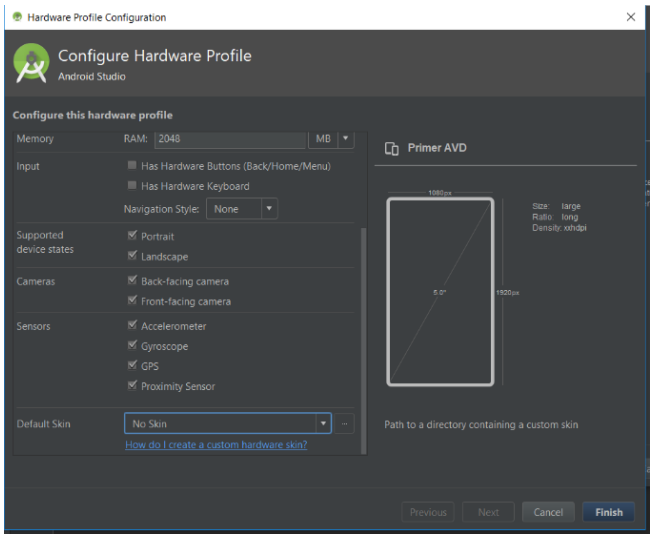


Figura 2.40: Configuración de hardware

Fuente: edición propia

Ya finalizada la creación del emulador regresará a la ventana donde seleccionamos un dispositivo, dentro de este seleccionar el emulador Primer AVD (Administrador de Dispositivos Virtuales) y dar clic en *Next*, dentro de esta ventana se mostrarán las imágenes del sistema que están disponibles para esta terminal, en este caso descargar la imagen de *Android oreo*, para su descarga solo dar *clic* en *Dowload* y aceptaremos los términos. Ya finalizada la descarga de la imagen, se podrá habilitar el botón para ir al siguiente paso.

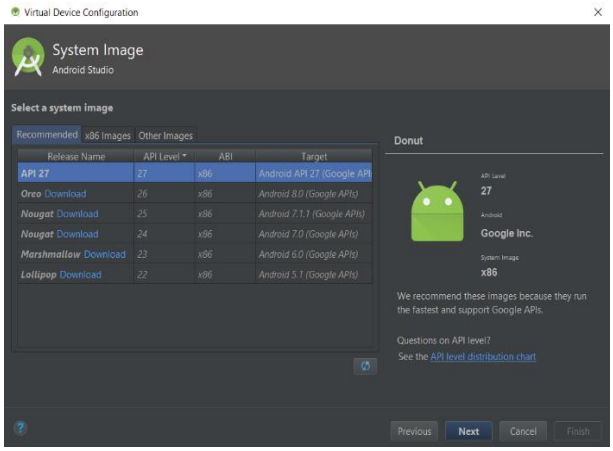


Figura 2.41: imágenes del sistema

Fuente: edición propia

En esta ventana revisaremos las configuraciones que acabos de realizar a nuestro emulador, en caso de estar todo correcto procederemos a finalizar su creación dando *clic* en *Finish.*

Se procede a ejecutar el proyecto, como se puede ver a continuación aparece la máquina virtual ya creada. Proceder a seleccionarla y a ejecutarla.

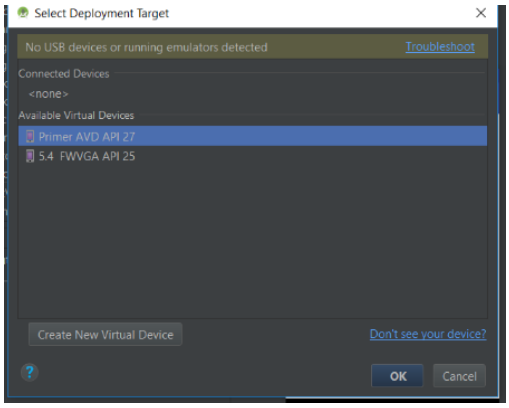


Figura 2.42: selección de la máquina virtual

Fuente: edición propia

Finalmente, así como luce la máquina virtual.

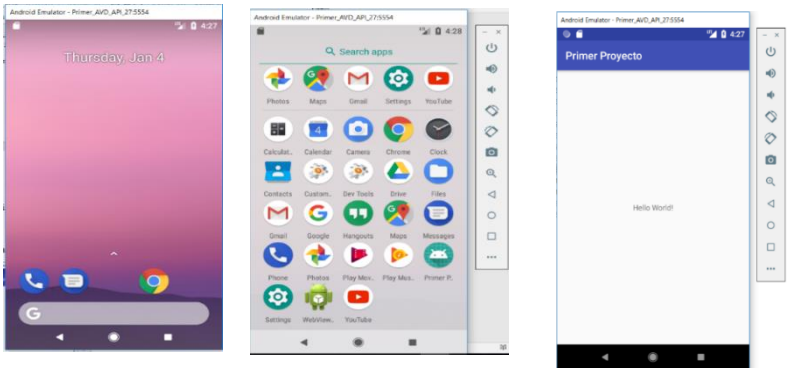


Figura 2.43: máquina virtual en ejecución

Fuente: edición propia

2.3. Estructura de proyectos en Android

Ya finalizada la creación del proyecto, Android muestra el *MainActivity.java* o actividad principal como se muestra en la figura 2.44.

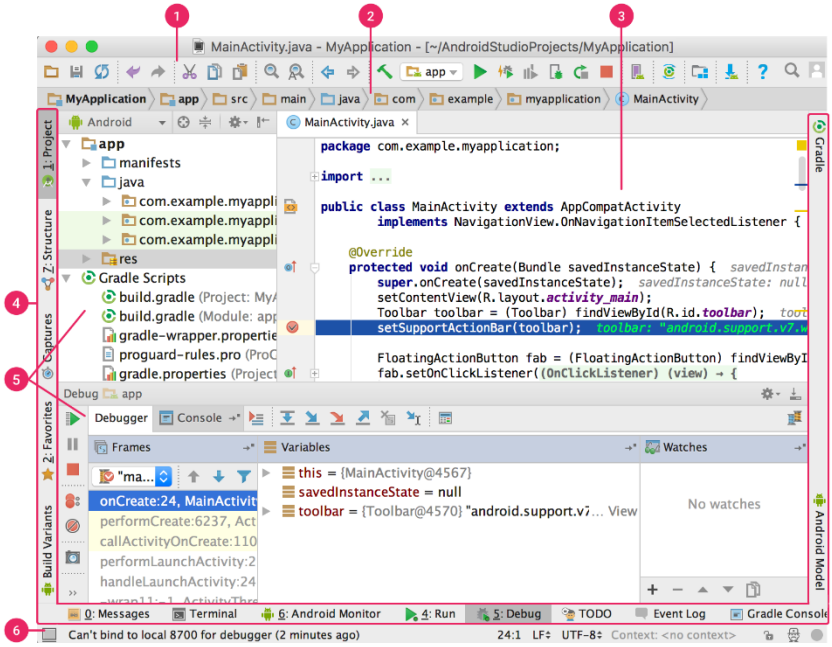


Figura 2.44: interfaz de usuario

Fuente: [32]

1. Barra de herramientas: dentro de esta barra podrá encontrar diversas acciones que nos ofrece Android Studio, entre las acciones más utilizadas permite la ejecución de nuestra *App*(*application*), correr las actualizaciones recientemente realizadas sin la necesidad de relanzar nuestra app, acceso directo al *AVD Manager* y el *SDK Manager*, acciones clásicas como son copiar, pegar y cortar.
2. Barra de navegación: Esta barra brinda su apoyo al momento de explorar un proyecto, proporciona una vista más simple de la ventana *Project*.
3. Ventana del editor: Dentro de esta ventana es podrá encontrar el área donde puede escribir y modificar el código, esta ventana puede variar según el tipo de archivo con el que se esté trabajando como puede ser el caso de los archivos XML(*extensible markup language*) los cuales ocupará el diseño de la interfaz de nuestra App.
4. Barra de la ventana de herramientas: se extiende alrededor de la parte externa de la ventana del IDE (Entorno de desarrollo integrado) y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.

5. Ventanas de herramientas: Dentro de esta ventana se tendrá acceso a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc.
6. Barra de estado: Dentro de esta barra podrá encontrar las notificaciones de advertencias o mensajes, estado del proyecto y el estado del IDE (Entorno de desarrollo integrado).

El proyecto Android se encuentra compuesto por varias carpetas como se muestra en la figura 2.45.

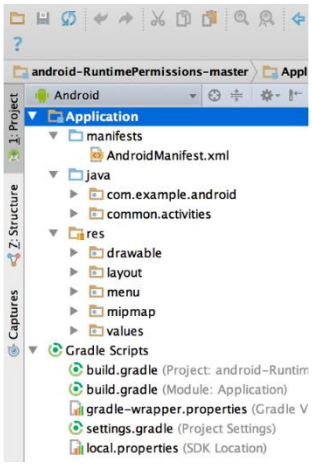


Figura 2.45: estructura de un proyecto *Android*

Fuente: [32]

Carpeta *manifests*: dentro de esta carpeta encontraras el archivo *AndroidManifest.xml*, este archivo contiene la configuración principal de la aplicación Android, dentro de este archivo podrá encontrar declarado las activities que ha creado, los permisos como son acceso a internet, permiso de escritura y lectura dentro de nuestro dispositivo, uso de los sensores y servicios.

Carpeta *java*: dentro de esta carpeta encontrará todas las clases correspondientes a las activities y clases propias.

Carpeta *res*: dentro de esta carpeta encontrará todos los recursos para el diseño de la interfaz, dentro de ella encontrará las siguientes carpetas:

Carpeta *drawable*: dentro de esta carpeta ubicará todos los archivos correspondientes a las imágenes he iconos propios que el *IDE(*Entorno de desarrollo integrado*)* proporciona.

- Carpeta *layout*: dentro de esta carpeta encontrara todos los archivos xml que corresponde a cada una de las activities, dentro de estas activities podrá comenzar a diseñar la interfaz de la aplicación con las distintas herramientas que puede usar como son *RecyclerView*, Botones, Cajas de texto, etc.
- Carpeta *menú*: dentro de esta carpeta encontrara los archivos xml(*extensible markup language*) correspondientes a los menús de la aplicación.
- Carpeta *values*: dentro de esta carpeta encontramos los siguientes archivos.
- *xml*: archivos string.xml: usado para definir las cadenas de caracteres.
- *colors.xml*: para definir los colores principales de la aplicación.
- *styles.xml*: usado para definir los estilos y temas.

2.4. Fases para el diseño de aplicaciones móviles

Fase 1: Planeación

Historia de usuarios

Para el desarrollo de esta primera fase, la metodología XP(*Extreme programming*) siguiere la implementación de plantillas llamadas “Historia de usuarios”, las historias de usuarios son redactadas únicamente por el cliente, funcionan para especificar los requisitos que debe cubrir el software a desarrollar, la finalidad de implementar este tipo de plantillas es con el fin de evitar la creación de varios documentos que son formales y reducir tiempos de administración. Cada historia debe ser redactada lo suficientemente comprensible para que los desarrolladores puedan implementar un plan de entregas de prototipos.

La plantilla que se estará utilizando para la creación de las historias de usuarios se muestra a continuación en el cuadro 2.1, cada uno de los campos se explica a continuación.

Historia de Usuario	
Numero: permite identificar a una historia de usuario.	Nombre : describe de manera general a una historia de usuario
Usuario: persona que utilizara la funcionalidad del sistema descrita en la historia de usuario	Iteración asignada: número de iteraciones, en que el cliente desea que se implemente una historia de usuario
Prioridad en negocios: grado de importancia que el cliente asigna a una historia de usuario.	Puntos estimados: número de semanas que se necesitara para el desarrollo de una historia de usuario
Riesgo en desarrollo: valor de complejidad que una historia de usuario representa al equipo de desarrollo	Programador responsable: persona encargada de programar cada historia de usuario.
Descripción: información detallada de una historia de usuario	
Observación: campo opcional utilizado para aclarar, si es necesario, el requerimiento descrito de una historia de usuario.	

Cuadro 2.1: formato historias de usuario

Fuente: [35]

Historia de Usuario	
Numero: 1	Nombre : inicio de sesión para los operadores
Usuario: operador de transporte	Iteración asignada: 2
Prioridad en negocios: Alto (Alto/Medio/Bajo)	Puntos estimados: 1
Riesgo en desarrollo: medio (Alto/Medio/Bajo)	Programador responsable: Salathiel Méndez Sánchez
Descripción: para que los operadores puedan acceder dentro de la app será necesario que se registren con su nombre y su apellido paterno, para evitar el intercambio de teléfonos entre ellos será necesario que se ligue un operador especifico por teléfono.	
Observación:	

Cuadro 2.2: ejemplo de historia de usuario

Fuente: edición propia

Una vez redactada la historia del usuario se procede a entregarse al desarrollador para proceder a evaluarla y proceder a asignarle un costo, es decir el tiempo que el desarrollador estima que se tardara en realizar, en caso de que la historia redactada por el cliente sea demasiado compleja para el desarrollador se procede a dividir esa historia más pequeña y proceder a evaluarlas de nuevo. Este proceso se ilustra en la figura 2.46.

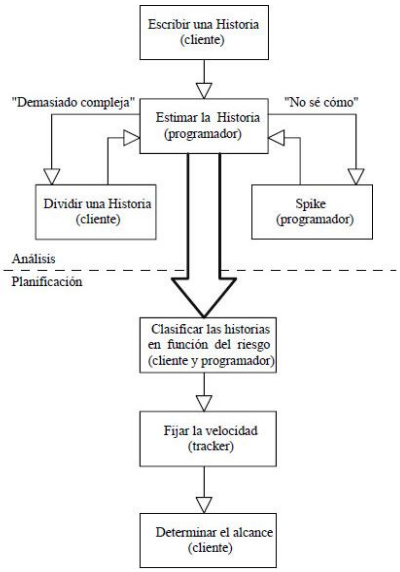


Figura 2.46 evaluación de historia de usuario

Fuente: [36]

Plan de Entregas

Ya realizadas y evaluadas las historias de usuario se procedió a crear el calendario de trabajo, este calendario servirá como el cronograma de actividades que se estarán llevando a cabo por cada historia de usuario en los lapsos de tiempos ya planificados.

Calendario de Trabajo	
Semana 2-11 de Agosto	
Jueves 3	Análisis de requerimientos
Viernes 4 a jueves 10	Desarrollo
Viernes 11	Entrega y Retroalimentación con el Usuario
Semana 14-21 de Agosto	
Lunes 14	Análisis de requerimientos
Martes 15- viernes 18	Modificaciones y Desarrollo
Lunes 21	Entrega y retroalimentación con el Usuario

Cuadro 2.3: formato de calendario de trabajo

Fuente: edición propia

Plan de iteración

Se estableció un plan de iteración por cada historia de usuario propuesta.

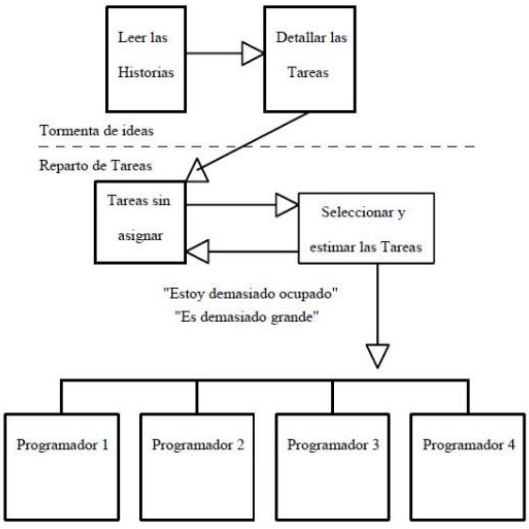


Figura 2.47. plan de iteración de historias de usuario

Fuente: [36]

Reuniones

Se establecieron reuniones diarias para poder evaluar los avances que se han ido efectuando a lo largo del desarrollo del proyecto y expresar inconvenientes con el desarrollo de esta, durante las reuniones se expusieron ciertos detalles e inconvenientes dentro de la empresa, para ello se enlista los problemas y soluciones planteadas dentro del equipo de desarrollo.

Problemática:

- Mal diseño de la base de datos con la que cuentan actualmente la empresa.
- Falta de relación entre las tablas.
- Ambigüedad.
- Tablas que no se están ocupando.
- Falta de normalización de la base de datos.

Soluciones propuestas:

- Nuevo diseño de la base de datos.

Fase 2: Diseño

Como se mencionó anteriormente, El diseño XP(*Extreme programming*) sigue rigurosamente el principio MS (mantenlo sencillo).se es recomendable tratar de tener todos nuestros diseños lo más simples y sencillos. Se procura hacer todo lo menos complicado posible, a la larga costara menos esfuerzo tiempo y esfuerzo su desarrollo, para ver el diseño de las interfaces, diseño propuesto de la base de datos, diseño de diagrama de clases véase en el capítulo IV. Para esta fase XP (*Extreme programming*) propone el uso de las tarjetas CRC (clase-responsabilidad-colaborador) en base a las historias de usuario esto con la finalidad de tener una mejor representación de las clases.

Tarjetas CRC	
Nombre de la clase: nombre de la clase al cual hace referencia	
Responsabilidades: atributos y operaciones de la clase	**Colaboradores:** clases que colaboran con la clase citada en la tarjeta

Cuadro 2.4: formato tarjetas CRC

Fuente:[35]

Tarjetas CRC	
Nombre de la clase: MainActivity	
Guardar el nombre y apellido del operador	
Verificar valides de los datos introducidos por el operador	
Devolver Imei, IdPersonal, nombre y apellido	

Cuadro 2.5: tarjeta CRC *MainActivity*

Fuente: edición propia

Tarjetas CRC	
Nombre de la clase: Menú	
Recibir coordenadas GPS y mandarlas al servidor	
Recibe Imei, IdPersonal, nombre y apellido	MainActivity
Presentar opción de asistencia y porteo	
Enviar IdPersonal, nombre y apellido	

Cuadro 2.5: tarjeta CRC m*enú*

Fuente: edición propia

Tarjetas CRC	
Nombre de la clase: Asistencia	
Recibe IdPersonal, nombre y apellido	Menu
Captura la hora que se registró la asistencia	

Cuadro 2.6: tarjeta CRC asistencia

Fuente: edición propia

Tarjetas CRC	
Nombre de la clase: Remisiones	
Recibe IdPersonal	Menu
Presenta una lista de remisiones	
Presenta una ventana donde se selecciona una opcion	
Presenta una ventana donde se captura un folio de remisión	
Envía el folio remisión capturado	

Cuadro 2.7: tarjeta CRC remisiones

Fuente: edición propia

Tarjetas CRC	
Nombre de la clase: Devolución	
Recibe folio de la remisión	Remisiones
Recibe la cantidad de presentaciones que fueron devueltas	
Recibe observación de la devolución	
Verifica que las cantidades capturas no sobrepasen a las que se cuentan asignadas	
Presenta una ventana de confirmación al usuario	

Cuadro 2.8: tarjeta CRC devolución

Fuente: edición propia

Fase 3: Codificación

El cliente es una parte más del equipo de desarrollo; su presencia es indispensable en las distintas fases de XP(*Extreme programming*). A la hora de codificar una historia de usuario su presencia es aún más necesaria. No olvidemos que los clientes son los que crean las historias de usuario y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario el cliente debe especificar detalladamente lo que ésta hará y también tendrá que estar presente cuando se realicen los test que verifiquen que la historia implementada cumple la funcionalidad especificada. La codificación debe hacerse ateniendo a

estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad. [37]

XP (*Extreme programming*) recomienda la programación en parejas para esta fase, sin embargo, el presente proyecto se está siendo desarrollado por una sola persona. Véase la codificación en el capítulo IV.

Fase 4: Pruebas

Una vez finalizados todos los módulos y sus procesos, se procede a llevar a cabo lo que son las pruebas, con la finalidad de verificar que el software esté funcionando correctamente o para dar a conocer alguna falla que no se detectó durante el desarrollo. Para ellos se enlistan las siguientes pruebas que se efectuaron:

- Pruebas de validación: Este tipo de prueba sirve para verificar que el sistema no permita ingresar valores que no son los correspondientes a los que se están solicitando dentro de la interfaz.
- Pruebas de usabilidad: Este tipo de prueba ayuda a determinar si la aplicación es lo suficientemente intuitiva para que el usuario pueda navegar dentro de ella sin problema alguno, es decir, que logre identificar para que sirve cada elemento.
- Pruebas de compatibilidad: Este tipo de prueba sirve para verificar que la aplicación pueda ser ejecutada sin problemas en distintos dispositivos Smartphone.

Pruebas de aceptación	
Código: N° único, permite identificar la prueba de aceptación	**N° historia de usuario:** numero único que identifica a la historia de usuario
Historia de usuario: nombre que indica de manera general la descripción de la historia de usuario	
Condiciones de ejecución: condiciones previas que deben cumplirse para realizar la prueba de aceptación	
Entrada / pasos de ejecución: pasos que siguen los usuarios para probar la funcionalidad de la historia de usuario	
Resultados de la prueba: respuestas del sistema que el cliente espera, después de haber ejecutado una funcionalidad	
Evaluación de la prueba: nivel de satisfacción del cliente sobre la respuesta del sistema. Los niveles son: aprobado y no aprobado	

Cuadro 2.9: formato pruebas de aceptación

Fuente:[35]

Pruebas de aceptación	
Código: 1	**N° historia de usuario:** 4
Historia de usuario: inicio de sesión para los operadores	
Condiciones de ejecución: cada operador escribirá su nombre y apellido y podrá acceder a su perfil correspondiente.	
Entrada / pasos de ejecución: Llenar el formulario, escribir nombre y apellido paterno Pulsar el botón iniciar	
Resultados de la prueba: acceso a la funciones de la aplicación móvil para elaborar sus actividades correspondientes	
Evaluación de la prueba: aprobado	

Cuadro 2.10: prueba de aceptación historia de usuario 4

Fuente: edición propia

Pruebas de aceptación	
Código: 2	**N° historia de usuario:** 5
Historia de usuario: Seguridad en el Inicio de sesión para los operadores	
Condiciones de ejecución: cada operador estará enlazado a un único teléfono al momento de registrarse	
Entrada / pasos de ejecución: Llenar el formulario, escribir nombre y apellido paterno Pulsar el botón iniciar	
Resultados de la prueba: se vinculó el teléfono al operador y no se puede acceder con otro operador	
Evaluación de la prueba: aprobado	

Cuadro 2.11: prueba de aceptación historia de usuario 5

Fuente: edición propia

Pruebas de aceptación	
Código: 3	**N° historia de usuario:** 6
Historia de usuario: ventana de Menú	
Condiciones de ejecución: cada operador podrá seleccionar una de las dos opciones disponibles para entrar a la ventana correspondiente	
Entrada / pasos de ejecución: Pulsar uno de los dos botones, asistencia o porteo	
Resultados de la prueba: acceso a la funciones de cada ventana	
Evaluación de la prueba: aprobado	

Cuadro 2.12: prueba de aceptación historia de usuario 6

Fuente: edición propia

Pruebas de aceptación	
Código: 4	**N° historia de usuario:** 7
Historia de usuario: GPS	
Condiciones de ejecución: al momento de entrar a la ventana menú se comienza a ejecutar el rastreo GPS	
Entrada / pasos de ejecución: Llenar el formulario de la ventana MainActivity y acceder a la ventana Menú	
Resultados de la prueba: activación del rastreo GPS	
Evaluación de la prueba: aprobado	

Cuadro 2.13: prueba de aceptación historia de usuario 7

Fuente: edición propia

Pruebas de aceptación	
Código: 5	**N° historia de usuario:** 8
Historia de usuario: ventana Asistencia	
Condiciones de ejecución: cada operador podrá marcar su hora de entrada y salida	
Entrada / pasos de ejecución: Pulsar el botón asistencia	
Resultados de la prueba: Se logró ingresar la asistencia	
Evaluación de la prueba: aprobado	

Cuadro 2.14: prueba de aceptación historia de usuario 8

Fuente: edición propia

Pruebas de aceptación	
Código: 6	**N° historia de usuario:** 9
Historia de usuario: Ventana lista de remisiones	
Condiciones de ejecución: cada operador podrá visualizar las remisiones que tiene a su cargo	
Entrada / pasos de ejecución: Pulsar el botón porteo Seleccionar cualquier elemento de la lista para ingresar su folio o abrir la siguiente ventana	
Resultados de la prueba: Se logró mostrar la lista de remisiones y hacer las correspondientes inserciones	
Evaluación de la prueba: aprobado	

Cuadro 2.15: prueba de aceptación historia de usuario 9

Fuente: edición propia

Pruebas de aceptación	
Código: 7	**N° historia de usuario:** 10
Historia de usuario: Ventana Devoluciones con pieza	
Condiciones de ejecución: cada operador podrá capturar las piezas devueltas	
Entrada / pasos de ejecución: Llenar los formularios que se mostraran al ingresar dentro de la ventana Pulsar el botón enviar Confirmar	
Resultados de la prueba: Se logró mostrar un formulario que contenga únicamente las piezas asignadas a esa remisión y proceder a enviarlas.	
Evaluación de la prueba: aprobado	

Cuadro 2.16: prueba de aceptación historia de usuario 10

Fuente: edición propia

2.5. Base de datos para dispositivos móviles

Una base de datos es aquella entidad que nos sirve como un almacén en la cual podremos guardar grandes cantidades de información de una manera estructurada y que a su vez puede ser consultada y reutilizada fácilmente por los usuarios. Las bases de datos pueden ser locales, esto quiere decir que será utilizada por un único usuario en su equipo de cómputo.

Los sistemas gestores des base de datos son aquellos programas que son capaces de acceder a los datos que han sido almacenados de una manera rápida y estructurada, esto a su vez les permite a los usuarios realizar diferentes tipos de

operaciones, entre las que encontramos las cuatro básicas: consultar datos, insertar datos, modificar datos y borrar datos.

Una base de datos móvil es aquella que puede ser instala dentro de nuestro dispositivo móvil, en la cual el usuario no necesita de una conexión a internet para poder almacenar o consultar su información.

SQLite es un motor de base de datos SQL(*Structured Query Language*) incorporado. El código para *SQLite* es de dominio público y, por lo tanto, es gratuito para cualquier uso, comercial o privado. A diferencia de la mayoría de las otras bases de datos *SQL*, SQLite no tiene un proceso de servidor por separado. *SQLite* lee y escribe directamente en archivos de disco ordinarios. Una base de datos SQL(*Structured Query Language*) completa con múltiples tablas, índices, disparadores y vistas, está contenida en un solo archivo de disco. El formato de archivo de la base de datos es multiplataforma: puede copiar libremente una base de datos entre sistemas de 32 bits y de 64 bits. Estas características hacen que *SQLite* sea una opción popular como formato de archivo de aplicación. [38]

Figura 2.48: logo *SQLite*
Fuente:[38]

Características de SQLite: [39]

- La base de datos completa se encuentra en un solo archivo.
- Puede funcionar enteramente en memoria, lo que la hace muy rápida.
- Tiene un *footprint* menor a 230KB.
- Es totalmente auto contenida (sin dependencias externas).
- Cuenta con librerías de acceso para muchos lenguajes de programación.
- Soporta texto en formato UTF-8(*Unicode Transformation Format*) y UTF-16(*Unicode Transformation Format*), así como datos numéricos de 64 bits.

Soporta funciones SQL (*Structured Query Language*) definidas por el usuario (UDF).

- El código fuente es de dominio público y se encuentra muy bien documentado.

En conclusión Android es un sistema operativo móvil que aún tiene mucho por ofrecernos, tanto para usuarios que simplemente lo utilizan día a día para realizar sus actividades cotidianas como para nosotros los desarrolladores de aplicaciones móviles, *Google* se ha encargado de ofrecernos un entorno de desarrollo oficial que nos brinda una gran cantidad de herramientas para la creación de *Apps*(*application*), así como herramientas de diseño como lo es *Material Desing*, incluso la facilidad de poder interactuar con los distintos sensores con los que cuenta un dispositivo móvil, esto brinda al desarrollador gran libertar para crear aplicaciones que puedan interactuar con el medio que rodea al usuario final.

Como todo *software* a desarrollar, se es necesario seguir una serie de pasos para asegurar el correcto funcionamiento y desarrollo del proyecto, es por ello que la utilización de una metodología que sea fácil de adaptarse a cambios repentinos en la orientación del proyecto así como lo es XP(*Extreme programming*), brinda a nosotros como desarrolladores la facilidad de implementar los cambios necesarios en el díselo o en el código sin la necesidad de tener que reestructurar toda la funcionalidad del proyecto al igual que la readaptación de cada uno de los documentos que se deben ir generando a lo largo del desarrollo.

3. Estado del arte

3.1. Aplicaciones que ya existen y cubren lo que se pretende solventar

El rastreo para flotillas ha sido una solución para las empresas que cuentan con una gran cantidad de vehículos que realizan repartos y se necesitan movilizar en grandes distancias, el constante monitoreo de estos vehículos ayuda a las empresas a tener un mejor control de los transportistas y poder conocer cuáles son los tiempos de arribo de cada una de las unidades en movimiento.

La localización vehicular proporciona un mejor habito de conducción y mejora la seguridad, los chóferes al saber que están siendo monitorizados en todo momento propicia que estos se enfoquen únicamente a la conducción evitando que se hagan desvíos no planeados en la ruta establecida, robo del cargamento, reducción de tiempos, esto al conocer la ruta que se está siguiendo ayuda a conocer si está tomando una ruta eficiente o ineficiente, conocer si los vehículos se están utilizando para usos no autorizados por parte de la empresa.

Muchos son los beneficios de contar con un sistema de rastreo satelital instalado en sus vehículos o unidades, entre los principales destacan: [40]

- Aumento de clientes al tener la carga monitorizada, es una seguridad adicional para los clientes saber que sus cargas se encuentran vigiladas
- Control total sobre la logística de rutas y entregas
- Disminución de viajes y kilómetros innecesarios
- Disminuye las primas del seguro
- Incremento de la productividad y rentabilidad, los operadores al saber que son monitorizados se enfocan a trabajar
- Manejo más seguro al tener control sobre la velocidad del vehículo, se tienen alertas en caso de un exceso de velocidad
- Paro de motor vía internet en caso de robo de la unidad además de saber su ubicación

- Permite asignar nuevo trabajo a vehículos que se encuentren cercanos.
- Permite auditar la conducta del conductor, mediante un histórico de ruta, velocidad, trayectoria y tiempos
- Permite la creación de geocercas, que alertan cuando el vehículo sale de la zona delimitada por el cliente
- Reducción de gastos de combustible, se evitan los viajes de los operadores a otros lugares que no sean sus destinos
- Reducción de gastos de mantenimiento, al controlar las rutas de los vehículos, los operadores dejan de usarlos como propios y se enfocan a hacer sus entregas
- Reduce el estrés de los dueños de los vehículos (taxis, camiones, camionetas, etc.) ya que saben con exactitud la ubicación de sus unidades.

TSO Fleet Pro

Aplicación móvil de rastreo vehicular GPS (sistema de posicionamiento global) disponible para *Android* y *IOS*, desarrollada por la empresa *UTS* Sistemas, disponible para usuarios que ya son clientes de *TSO Movile*. Permite el rastreo de vehículos en tiempo real, así como mostrar información detallada de la actividad histórica.

Figura 3.1: logo *Fleet Pro*

Fuente: [41]

Características: [41]

- Acceda a una lista de todos sus vehículos y seleccione la unidad que desea rastrear.
- Visibilidad en tiempo real de la ubicación actual de sus vehículos en el mapa.
- 3 vistas de mapa diferentes (calle, híbrido, satélite).
- Haga clic en el icono del vehículo en el mapa para obtener información actualizada de su ubicación y estado.

- Seleccione un rango de fechas para ver las ubicaciones históricas en el mapa.
- Selección de vehículos múltiples en la lista de unidades.
- Sistema de Sensores, con Historial de Sensores, Filtros, Notificaciones.
- Capacidades de chat y mensajería a través de dispositivos Garmin.
- Gestión de órdenes de trabajo.
- Control remoto de sus unidades.
- Requiere suscripción

Fleetmatics Reveal

Aplicación disponible en Android y IOS, desarrollado por la empresa *Fleetmatics* para el seguimiento de flotas vehiculares, permite a los usuarios acceder a la información clave de su flota, datos en tiempo real y el rendimiento de cada vehículo.

Figura 3.2: logo *Reveal*
Fuente: [42]

La aplicación nativa de Android te brinda una funcionalidad completa, no solo la navegación web móvil:[42]

- Ubique rápidamente cualquier controlador en su fuerza de trabajo móvil o encuentre el técnico más cercano a un trabajo urgente.
- Manténgase en contacto con el desempeño contra sus puntos de referencia. con indicadores y cuadros de mandos.
- Reciba alertas y notificaciones de actividades en tiempo real en su teléfono.
- Despliegue hacia Route Replay para investigar incidentes en el campo.
- Crear una nueva Geofence desde cualquier lugar.

GPSMonitor

Aplicación disponible para *Android* y *IOS*, desarrollada como complemento de la plataforma de rastreo que lleva el mismo nombre, se pueden visualizar la información de las unidades que se están rastreando mediante una interfaz fácil de utilizar, la información de cada vehículo se encuentra disponible de manera resumida y con fácil acceso a los comandos para apagar y encender los vehículos de forma remota.

Figura 3.3: logo *GPSMonitor*

Fuente: [43]

Características: [43]

- Frecuencia de rastreo de 1 minuto cuando el vehículo está encendido
- Animación de recorrido.
- Plataforma *web* y móvil.
- Accesos y usuarios limitados.
- Alarma en tiempo real.
- Historial de 90 días.

En conclusión Tras haber presentado algunas de las distintas alternativas de rastreó vehicular que existen actualmente en el mercado, se determina que para el uso de estos servicios se es necesario contar con la afiliación a la empresa y pagar una cantidad de cuota por todos los servicios que brinda su aplicación, sin embargo, muchos de estos servicios para la empresa a la cual se desarrolla este proyecto parecen incensarios ya que solo buscan conocer la ubicación actualmente sin la necesidad de generación de algún tipo de reporte extra. Por ello nace la propuesta de incluir un módulo de rastreo GPS (sistema de posicionamiento global) dentro de su aplicación móvil, enfocándose únicamente en los aspectos que se consideran únicamente necesarios.

4. Metodología

4.1. Tipo de investigación

En esta investigación, se emplea el tipo teórica-aplicada, debido a que durante la primera etapa se realizó un análisis de la información obtenida a lo largo de una investigación seleccionando los datos más significativos relacionados con el tema a investigar, con la finalidad de añadir conocimiento nuevo al ya existente de manera que ayude a profundizar cada vez más dentro del problema a solventar, finalmente con los datos obtenidos se dio inicio con la fase de aplicación donde se procede a crear las soluciones a la problemática a partir de los conocimientos adquiridos previamente.

4.2. Alcance de la investigación

En esta investigación, se emplea el alcance correlacional, puesto que en las hipótesis planteadas se especifica que se va a evaluar la relación que existe entre la variable desarrollo de una aplicación móvil (variable independiente) y las variables reducir tiempo, hacer más eficiente el proceso de entregas y conocer la ubicación actual de cada vehículo de reparto (variables dependientes), tomando como universo de estudio a todos los trabajadores que se encuentran en la empresa actualmente, y considerando como muestra únicamente a los diez empleados que operan los vehículos de reparto.

4.3. Diseño de investigación

La presente investigación muestra un diseño no experimental ya que no se lleva a cabo la manipulación de las variables, por el contrario, solo se registrarán y observarán las variables de estudio para analizar su comportamiento. Considerando el análisis de los procesos que realiza la empresa El Shaddai para la distribución de sus productos, esta investigación busca evaluar los tiempos empleados para cada

uno de los procesos, así como determinar los beneficios que puede brindar la implementación de una Aplicación móvil. Asimismo, se considera un esquema de recolección única de datos, de tal forma que es un diseño transeccional categorizado como correlacional- causales, ya que son los que describen relaciones entre dos o más variables en un momento determinado.

4.4. Cronograma de investigación

La investigación se realizó en buenos términos contando con la mejor disposición de la empresa; a continuación, se muestra la distribución de tiempo para su realización.

CRONOGRAMA

ACTIVIDADES		Agosto				Septiembre				Octubre				Noviembre				Diciembre				Enero			
		Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4
Generación de Idea	P	X	X																						
	R		X	X																					
Planteamiento del problema	P			X	X	X																			
	R				X	X	X	X																	
Revisión de la literatura y desarrollo de marco teórico	P				X	X	X	X																	
	R						X	X	X	X															
Elaboración de hipótesis y definición de variables	P						X	X	X	X															
	R									X	X	X													
Selección del diseño de investigación	P								X	X	X	X													
	R									X	X	X	X												
Definición y selección de muestra	P										X	X	X	X											
	R											X	X	X											
Análisis de los datos	P											X	X	X	X										
	R												X	X	X										
Elaboración del reporte de resultados	P															X	X	X	X			X	X	X	X
	R															X	X	X	X			X	X	X	X

Figura 4.1: cronograma de investigación.

Fuente: edición propia

4.5. Instrumentos de recolección y medición

La recolección de datos se refiere al uso de una gran diversidad de técnicas y herramientas que pueden ser utilizadas por el analista para desarrollar los sistemas de información, los cuales pueden ser la entrevistas, la encuestas, el cuestionario, la observación, el diagrama de flujo y el diccionario de datos. Con la finalidad de buscar información que será útil a una investigación en común. [45]

La técnicas de recolección de datos empleada es la observación, misma que se determinó fuera dirigida a las horas de envío y recepción de información a los empleados encargados del área de porteo, esto con finalidad de conocer el tiempo que le tomaba el operador de los vehículos repartidores en mandar la información actualizada correspondiente a las entregas que realizaron, esta observación se mantuvo durante dos semanas para poder identificar de manera puntual el flujo de la información generada por los operadores.

La entrevista también se empleó a través de una serie de preguntas realizadas a los operadores que serían los encargados de utilizar la aplicación móvil en el trascurso de su horario laboral en donde el objetivo de la entrevista fue conocer la experiencia que estas personal tiene con la utilización de Apps, considerando la entrevista al 100% de los operadores tomando en cuenta que sólo son 10 los operadores que trabajan en la empresa.

Se realizó una encuesta a los operadores que utilizaron la aplicación móvil con la finalidad de conocer su nivel de satisfacción que estos tuvieron al momento de interactuar con la aplicación móvil y así determinar los ajustes que se deben realizar, tomando en cuenta únicamente a los 10 operadores.

En conclusión, a lo largo de este capítulo se aborda la metodología de investigación que se empleó en el desarrollo de este proyecto, se estableció el tipo de investigación que fue empleada, el cronograma indicando el tiempo planeado y el real empleado en la investigación y finalmente las técnicas de recolección de datos. La finalidad de este capítulo es identificar cada uno de los elementos antes mencionados que componen la presente investigación, es importante que antes de comenzar a investigar y proceder al desarrollo, hay que delimitar el objeto de estudio y establecer los límites dentro de la investigación para ayudar a tener mejor enfoque en el área a investigar, finalmente los instrumentos de recolección de datos como las entrevistas y encuestan permiten conocer el porcentaje de aceptación que presenta proyecto, brindando respuesta a las hipótesis planteadas.

5. Desarrollo e implementación

5.1. Recolección e implementación de la App

Como mencionamos anteriormente el desarrollo de esta aplicación está bajo la metodología XP(*Extreme programming*),para la recolección y análisis de requerimientos se procedió a generar las Historia de Usuarios, las historias de usuarios son redactadas únicamente por el cliente, funcionan para especificar los requisitos que debe cubrir el software a desarrollar.

Historia de Usuario	
Numero: 1	Nombre: Propuesta de una aplicación móvil
Usuario: Operador de Transporte	**Iteración asignada:** 2
Prioridad en negocios: alto (Alto/Medio/Bajo)	**Puntos estimados:** 2
Riesgo en desarrollo: medio (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: propuesta de una aplicación móvil para teléfonos con Android para poder llevar el registro de las asistencia de los operadores	
Observación:	

Cuadro 5.1: historia de usuario 1

Fuente: edición propia

Historia de Usuario	
Numero: 2	**Nombre:** Propuesta de una aplicación móvil
Usuario: operador de Transporte	**Iteración asignada:** 1
Prioridad en negocios: alto (Alto/Medio/Bajo)	**Puntos estimados:** 2
Riesgo en desarrollo: medio (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: propuesta de una aplicación móvil para teléfonos con Android para poder capturar los folios y las piezas de las devoluciones de cada remisión	
Observación:	

Cuadro 5.2: historia de usuario 2

Fuente: edición propia

Historia de Usuario	
Numero: 3	**Nombre :** Ventana de inicio de sesión
Usuario: operador de Transporte	**Iteración asignada:** 1
Prioridad en negocios: alto (Alto/Medio/Bajo)	**Puntos estimados:** 1
Riesgo en Desarrollo: medio (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: diseñar una venta para que los operadores logren iniciar sesión y entrara a la aplicación	
Observación:	

Cuadro 5.23 historia de usuario 3

Fuente: edición propia

Historia de Usuario	
Numero: 4	**Nombre:** inicio de sesión para los operadores
Usuario: operador de Transporte	**Iteración asignada:** 2
Prioridad en negocios: alto (Alto/Medio/Bajo)	**Puntos estimados:** 1
Riesgo en desarrollo: medio (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: para que los operadores puedan acceder dentro de la app será necesario que se registren con su nombre y su apellido paterno.	
Observación:	

Cuadro 5.4: historia de usuario 4

Fuente: edición propia

Historia de Usuario	
Numero: 5	**Nombre :** Seguridad en el Inicio de sesión para los operadores
Usuario: operador de Transporte	**Iteración asignada:** 2
Prioridad en negocios: alto (Alto/Medio/Bajo)	**Puntos estimados:** 2
Riesgo en Desarrollo: medio (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: Para evitar el intercambio de teléfonos entre ellos será necesario que se ligue un teléfono a un solo operador.	
Observación: en dado caso que un operador le proporcione su teléfono a otro operador para realizar alguna actividad relacionada a su trabajo, no se debe iniciar sesión con sus credenciales.	

Cuadro 5.5: historia de usuario 5

Fuente: edición propia

Historia de Usuario	
Numero: 6	**Nombre :** Ventana de Menú
Usuario: operador de Transporte	**Iteración Asignada:** 1
Prioridad en Negocios: medio (Alto/Medio/Bajo)	**Puntos estimados:** 1
Riesgo en Desarrollo: bajo (Alto/Medio/Bajo)	**Programador Responsable:** Salathiel Méndez Sánchez
Descripción: se debe diseñar una ventana intuitiva que le proporcione a los operadores las opciones de entrar a la ventana de asistencias o porteo.	
Observación:	

Cuadro 5.6: historia de usuario 6

Fuente: edición propia

Historia de Usuario	
Numero: 7	**Nombre :** GPS
Usuario: operador de transporte	**Iteración asignada:** 1
Prioridad en negocios: alto (Alto/Medio/Bajo)	**Puntos estimados:** 1
Riesgo en desarrollo: alto (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: al momento que un operador se registra con sus credenciales el teléfono debe comenzar a transmitir su posición GPS al servidor de la empresa.	
Observación:	

Cuadro 5.7 historia de usuario 7

Fuente: edición propia

Historia de Usuario	
Numero: 8	**Nombre :** ventana asistencia
Usuario: operador de transporte	**Iteración asignada:** 2
Prioridad en negocios: medio (Alto/Medio/Bajo)	**Puntos estimados:** 1
Riesgo en desarrollo: bajo (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: se debe diseñar una ventana para que el operador pueda marcar su hora de entrada, comida, entrada comida. Cuando este marcado que el operador regreso de su hora de comer se debe bloquear el botón.	
Observación: es importante que se bloque el botón para que el operador regrese a la bodega y marque su salida en el checador.	

Cuadro 5.8: historia de usuario 8

Fuente: edición propia

Historia de Usuario	
Numero: 9	**Nombre :** ventana lista de remisiones
Usuario: operador de transporte	**Iteración asignada:** 3
Prioridad en Negocios: medio (Alto/Medio/Bajo)	**Puntos estimados:** 2
Riesgo en Desarrollo: medio (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: se debe diseñar una ventana en la cual el operador pueda ver todas las remisiones que tengan asignadas a su cargo, al seleccionar una remisión se deberá desplegar un menú donde se pueda seleccionar una de las dos opciones siguientes. 1.-Devolucion sin piezas, 2.- Devolución con piezas.	
Observación: En el caso de ser devolución sin pieza permita ingresar el folio que se le otorga al operador , en caso contrario abrir la siguiente ventana.	

Cuadro 5.9: historia de usuario 9

Fuente: edición propia

Historia de Usuario	
Numero: 10	**Nombre :** ventana devoluciones con pieza
Usuario: operador de transporte	**Iteración asignada:** 3
Prioridad en Negocios: alto (Alto/Medio/Bajo)	**Puntos estimados:** 3
Riesgo en Desarrollo: alto (Alto/Medio/Bajo)	**Programador responsable:** Salathiel Méndez Sánchez
Descripción: se debe diseñar una ventana intuitiva en la cual le muestre al operador únicamente las piezas que trae asignada cada remisión a su cargo.	
Observación: Únicamente se tienen que mostrar las presentaciones que tiene asignadas esa remisión, no se debe de poder ingresar valores que no concuerden a los que se tienen capturados.	

Cuadro 5.10: historia de usuario 10

Fuente: edición propia

5.2. Diseño de la App

5.2.1. Casos de uso

Un caso de uso es una descripción de las actividades que deben realizarse para llevar a cabo un proceso. Representan las funciones que proporciona un sistema que son de valor para sus usuarios. Los diagramas de caso de uso nos ayudan a modelar los requisitos funcionales de nuestro sistema, de tal forma que veremos las relaciones que existen entre los requisitos (casos de uso) y los actores (que pueden ser personas u otros sistemas). [44].

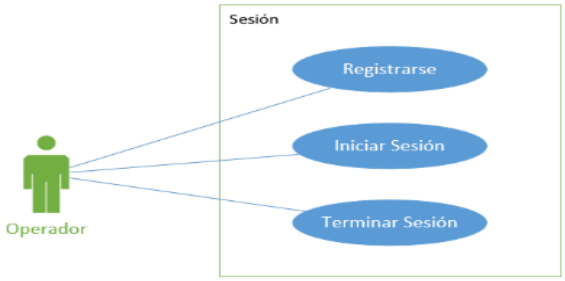


Figura 5.1: caso de uso 1

Fuente: edición propia

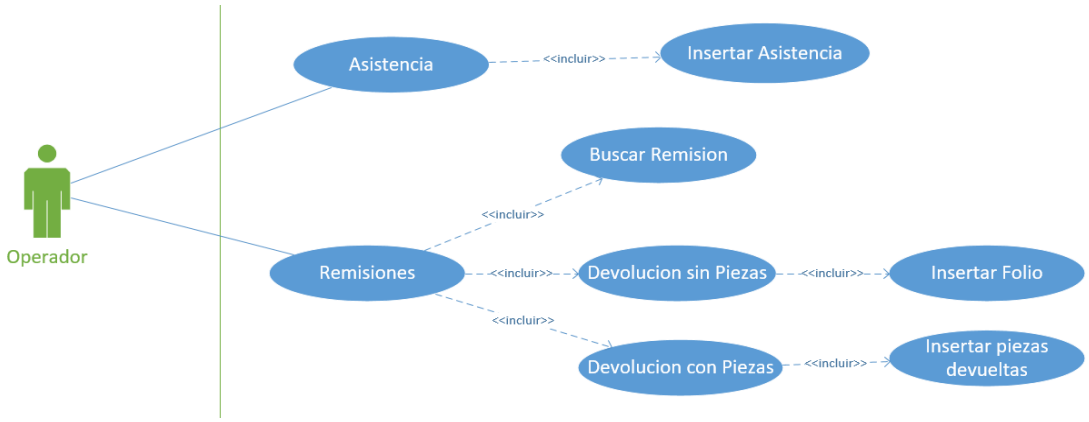


Figura 5.2: caso de uso 2

Fuente: edición propia

5.2.2. Diagrama de clases

Un diagrama de clases es una representación gráfica que sirve para representar la estructura de un sistema que será implementado utilizando un lenguaje orientado a objetos. Los diagramas de clases se realizan en la fase de diseño del software después de la fase de requisitos. La idea de estos diagramas es representar las clases que tendrá el sistema, así como su contenido y sus relaciones con otras clases.

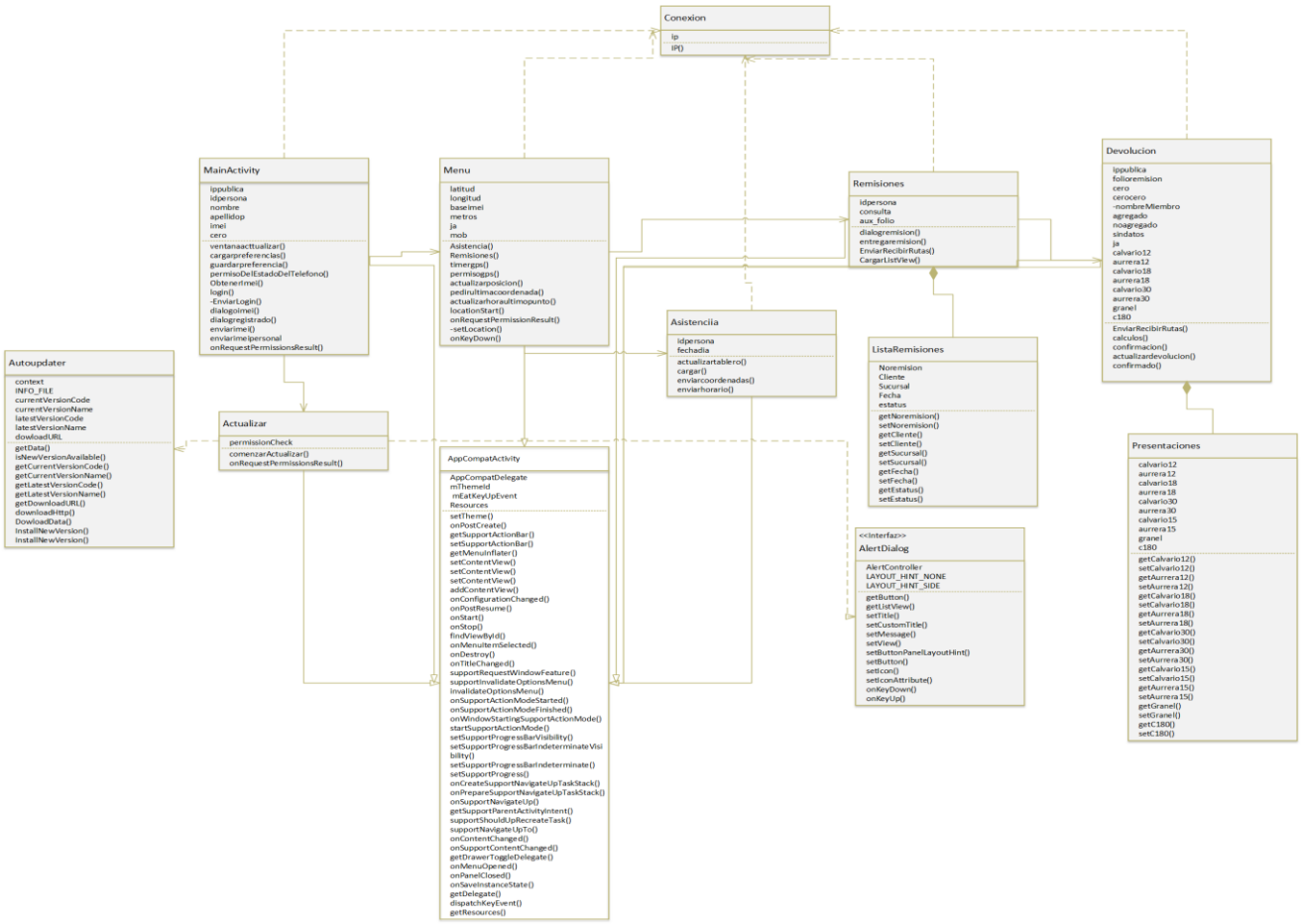


Figura 5.3: diagrama de clases

Fuente: edición propia

5.2.3. Diseño de vistas

La interfaz de usuario es el medio por la cual este pueda interactuar con su dispositivo móvil, mandar mensajes, compartir archivos, utilizando un conjunto de objetos gráficos para mostrar información y acciones disponibles.

Al iniciar la aplicación se muestra una pantalla como la siguiente figura. En esta pantalla se muestran los elementos necesarios: 2 *EditText* que funcionaran como medio para ingresar su nombre y apellido, un *Button* o botón de Entrar.

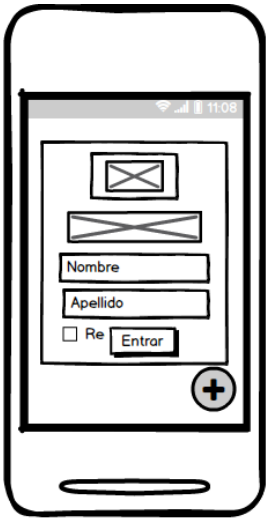


Figura 5.4: diseño de login

Fuente: edición propia

Tras iniciar sesión se muestra una pantalla como se muestra en la siguiente figura. Dentro de esta pantalla encontraremos 2 botones asistencia que nos redirigirá a la ventana de Asistencia, botón remisiones que nos redirigirá a la ventana donde se mostrará una lista de las remisiones a cargo del operador.

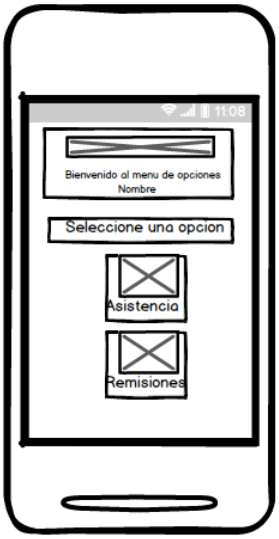


Figura 5.5: diseño menú

Fuente: Edición propia

La Tras iniciar sesión se muestra una pantalla como se muestra en la siguiente figura. Dentro de esta pantalla encontramos los siguientes elementos, *TextView* en el cual se mostrar el mensaje de bienviva, el nombre del operador, fecha actual,

títulos de las asistencias, un botón registrar asistencia al presionarlo marcara la asistencia del operador, botón flotante que al ser presionado desplegara una opción para poder actualizar el tablero de asistencias.

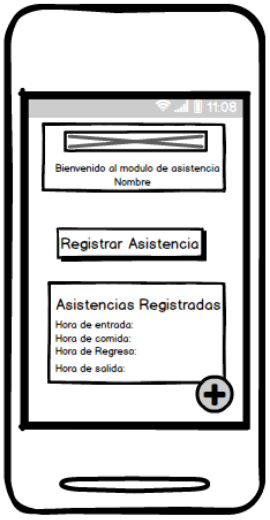


Figura 5.6: diseño asistencia

Fuente: edición propia

Tras presionar la opción remisiones nos redirigirá la ventana como se muestra en la siguiente imagen, dentro de esta ventana se encuentra listadas las remisiones que están a cargo del operador, al seleccionar un elemento de la lista mostrará un mensaje donde nos dará la opción de poder especificar si la acción que realizaremos. En la barra de menús de esta misma ventana se verá el icono de una lupa, al ser presionado nos mostrará una caja de texto la cual servirá para poder ingresar información de las remisiones para proceder a filtrarlas.

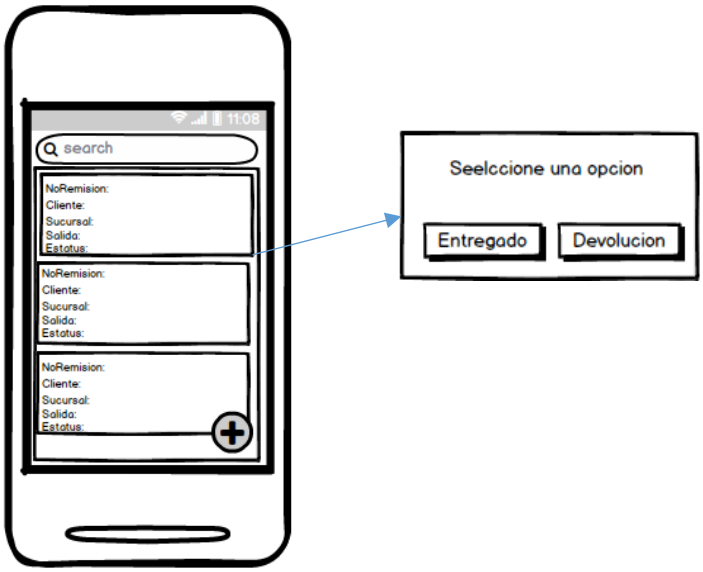


Figura 5.7: diseño de listo de remisiones

Fuente: edición propia

Al presionar el elemento entregado nos mostrara una ventana emergente como se muestra en la siguiente figura. Dentro de esta caja de texto se procede a ingresar el folio que se le proporciona al operador, en caso contrario y seleccionar devolución, se muestra una ventana como se muestra en la siguiente figura, dentro de esta pantalla contamos con caja de texto en la parte superior de la pantalla donde se procede a ingresar el folio que se ha proporcionado, en la parte inferior de la ventana se visualizara cajas de texto donde se podrá ingresar las devoluciones que se ha tenido de cada presentación, al final encontramos un botón que al ser presionado mandara la información al servidor de la empresa.

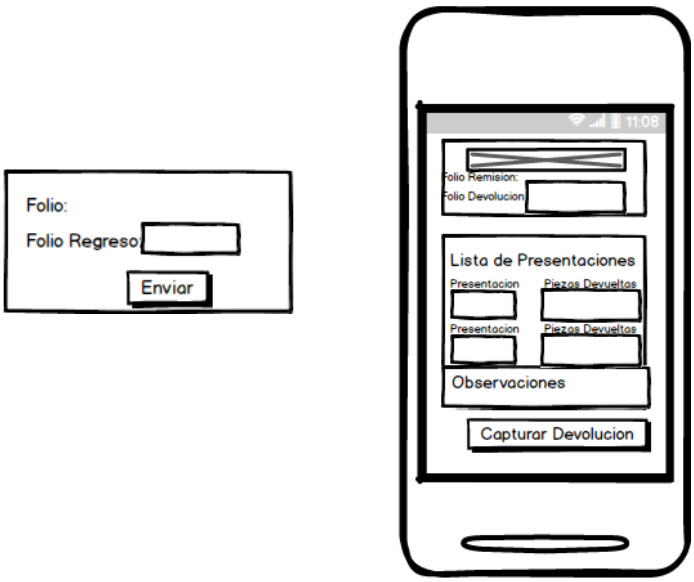


Figura 5.8: diseño devolución

Fuente: edición propi

5.3. Diseño y creación de la base de datos

Las bases de datos juegan un papel importante en la mayoría de las áreas donde se utilizan las computadoras, permitiendo almacenar grandes volúmenes de datos acerca de la empresa, los cuales son percibidos a través de los usuarios, de la misma manera la información obtenida de los datos almacenados debe estar en una forma que sirva para administrar, planear, controlar y tomar decisiones dentro de una organización. [45].

Actualmente la empresa cuenta con su propia de base datos que se encuentra en producción, a lo largo del desarrollo de este proyecto se encontraron inconvenientes en el diseño de la base de datos con la que cuentan, a continuación, se enlistan los inconvenientes que se recabaron:

- Tablas sin utilizar
- Tablas con campos que no se utilizan
- Tablas sin relaciones
- Ambigüedad

A pesar de que la empresa se encuentra trabajando bajo estas condiciones, se hizo la propuesta de la creación de un nuevo diseño de la base de datos dirigida

únicamente al módulo de devoluciones. Para realizar este diseño se procede a aplicar las tres reglas de normalización de base de datos.

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener. También se puede entender la normalización como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede. [46]

Regla	Descripción
Primera Forma Normal (1FN)	Incluye la eliminación de todos los grupos repetidos.
Segunda Forma Normal (2FN)	Asegura que todas las columnas que no son llave sean completamente dependientes de la llave primaria (PK).
Tercera Forma Normal (3FN)	Elimina cualquier dependencia transitiva. Una dependencia transitiva es aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave.

Figura 5.9: reglas de formalización

Fuente: [47]

Primera Forma Normal

La primera forma normal establece que no debe existir columnas repetidas dentro de nuestras tablas, toda aquella columna que se encuentre repetida debe de eliminarse y posteriormente colocarse en tablas separadas.

Segunda Forma Normal

La segunda forma normal establece que aquellos datos que no son considerados llave primaria deben depender por completo de la llave primaria.

Tercera Forma Normal

Una tabla está normalizada en esta forma si todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas. Comentamos anteriormente que una dependencia transitiva es aquella en la cual existen columnas que no son llave que dependen de otras columnas que tampoco son llave. [48]

Aplicando estas tres formas de normalización se procedió a crear el nuevo diseño de la base de datos propuesta, en la siguiente figura se muestra el diseño final de la base de datos.

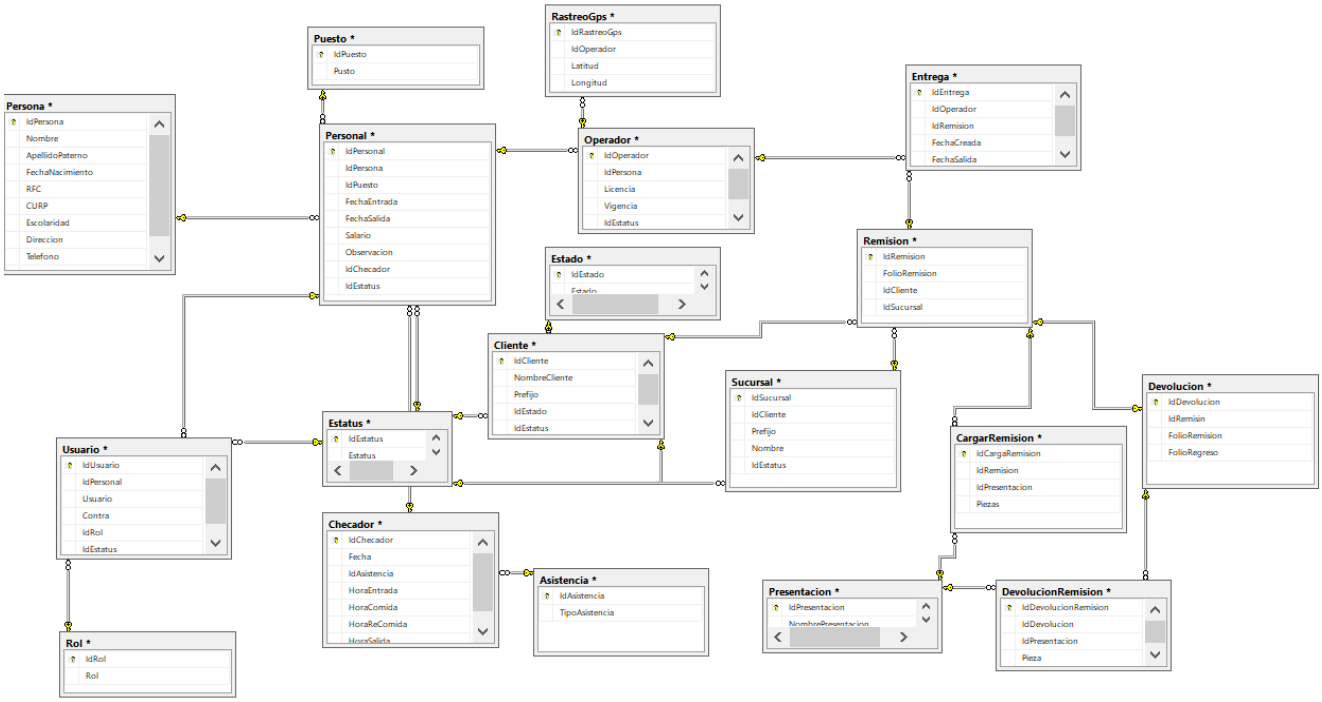


Figura 5.10: diseño propuesto de la base de datos

Fuente: edición propia

5.4. Codificación de App

Una vez que ya se han analizado los requisitos y se tiene el diseño de la aplicación se procede al desarrollo el código de la aplicación Android y de los archivos *.aspx* que nos servirán como medio de conexión con la base de datos.

Para poder iniciar sesión dentro de la aplicación se realizó el siguiente archivo llamado login.aspx el cual consiste en preguntar si el usuario que está ingresando sus credenciales esta dado de alta como chofer y si es un operador activo.

```
try
        {
       if (Request.QueryString["Nombre"] != "" && Request.QueryString["AP"] != "")
            {
                SqlCommand cmd = new SqlCommand("SP_ANDROID_LOGIN", con);
                cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.AddWithValue("@Nombre", Request.QueryString["Nombre"].Replace("_"," "));
                cmd.Parameters.AddWithValue("@App", Request.QueryString["AP"]);
                SqlDataAdapter da = new SqlDataAdapter(cmd);
```

```
                DataTable dt = new DataTable();
                da.Fill(dt);
                foreach (DataRow row in dt.Rows)
                {
                    string imei = "";
                    if (row[3].ToString() == "")
                    {
                        imei = "0";
                    }
                    else
                    {
                        imei = row[3].ToString();
                    }
 Response.Write("[\"" + row[0].ToString() + "\"," + "\"" + row[1].ToString() + "\"," +
"\"" + row[2].ToString() + "\"," + "\"" + imei + "\"]");
                }
            }
            else
            {
                Response.Write("No hay datos");
            }
        }
        catch
        {
            Response.Write("No hay datos");
        }
```

Una vez que se validó de que el usuario es un chofer se procede a registrar el dispositivo móvil a su registro en la base de datos, Para realizar el registro de los choferes y que estos tengan acceso a la aplicación se realizó el siguiente archivo registrotelefonousuario.aspx.

```
try
        {
if (Request.QueryString["IdPersona"] != "" && Request.QueryString["Imei"] != "")
            {
                SqlCommand cmd = new SqlCommand("SP_ANDROID_BUSCAR_IMEI", con);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@imei", Request.QueryString["Imei"]);
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                da.Fill(dt);
                DataRow row = dt.Rows[0];
                int cantidad =Convert.ToInt32(row[0].ToString());
                if (cantidad > 0)
                {
                    Response.Write("No Agregado");
                }
                else {
SqlCommand cmd1 = new SqlCommand("SP_ANDROID_REGISTRAR_USUARIO", con);
                    cmd1.CommandType = CommandType.StoredProcedure;
                    cmd1.Parameters.AddWithValue("@idpersona",
Request.QueryString["IdPersona"].Trim());
                    cmd1.Parameters.AddWithValue("@imei",
Request.QueryString["Imei"].Trim());
                    con.Open();
                    cmd1.ExecuteNonQuery();
                    con.Close();
                    Response.Write("Agregado");
```

```
            }
        }
        else
        {
            Response.Write("No Hay Nada");
        }
    }
    catch
    {
        Response.Write("Sin Datos");
    }
```

Para poder registrar las asistencias se creó el siguiente archivo InsertarAsistencia.aspx, el cual recibe el identificador del chofer.

```
try
    {
        if (Request.QueryString["IdPersona"] != "")
        {
            SqlCommand cmd = new SqlCommand("SP_APP_Insertar", con);
            cmd.CommandType = CommandType.StoredProcedure;
     cmd.Parameters.AddWithValue("@idFolio", Request.QueryString["IdPersona"]);
            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            Response.Write("Agregado");
        }
        else
        {
            Response.Write("No Agregado");
        }
    }
    catch
    {
        Response.Write("Sin Datos");
    }
```

Para realizar la petición de las remisiones que tiene asignado el chofer se realizó el siguiente archivo remisiones.aspx, consiste en pedir el identificador del chofer para obtener el listado de remisiones.

```
     if (Request.QueryString["IdPersona"] == "")
    {
        Response.Write("No se encontro");

    }
    else
    {
        try
        {
            SqlCommand cmd = new SqlCommand("SP_ANDROID_REMISIONES", con);
            cmd.CommandType = CommandType.StoredProcedure;
   cmd.Parameters.AddWithValue("@idpersona", Request.QueryString["IdPersona"]);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            da.Fill(dt);
```

```
                foreach (DataRow row in dt.Rows)
                {
Response.Write("[\"" + row[0].ToString() + "\"," + "\"" + row[1].ToString() + "\"," +
"\"" + row[2].ToString() + "\"," + "\"" + row[3].ToString() + "\"," + "\"" +
row[4].ToString() + "\"]");
                }
            }
            catch (Exception ex)
            {
                Response.Write("No existe");
            }
```

Para realizar las entregas que no cuentan con ninguna devolución se procedió a crear el siguiente archivo Devolucionsp.aspx, consiste en capturar el folio que se está ingresando y mandando las piezas como cero.

```
try
        {
            if (Request.QueryString["NoRemision"] != "" && Request.QueryString["Folio"]
!= "")
            {
                SqlCommand cmd = new SqlCommand("SP_IP_Capturar_Devoluciones", con);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@NoRemision",
Request.QueryString["NoRemision"].Trim());
                cmd.Parameters.AddWithValue("@Folio",
Request.QueryString["Folio"].Trim());
                cmd.Parameters.AddWithValue("@12CalPieza",cero );
                cmd.Parameters.AddWithValue("@12AuPieza",cero);
                cmd.Parameters.AddWithValue("@18CalPieza",cero);
                cmd.Parameters.AddWithValue("@18AuPieza",cero);
                cmd.Parameters.AddWithValue("@30CalPieza",cero);
                cmd.Parameters.AddWithValue("@30AuPieza",cero);
                cmd.Parameters.AddWithValue("@15CalPieza",cero);
                cmd.Parameters.AddWithValue("@15AuPieza",cero);
                cmd.Parameters.AddWithValue("@GranielKili",cero1);
                cmd.Parameters.AddWithValue("@180Pieza",cero1);
                con.Open();
                cmd.ExecuteNonQuery();
                con.Close();

                Response.Write("Agregado");
            }
            else
            {
                Response.Write("No Agregado");
            }
        }
        catch
        {
            Response.Write("Sin Datos");
        }
```

Para realizar la inserción de las remisiones que cuentan con piezas devueltas se procedió a crear el siguiente archivo Devolucionp.aspx

```
try
        {
```

```
            if (Request.QueryString["NoRemision"] != "" && Request.QueryString["Folio"]
!= ""&& Request.QueryString["CalPieza12"] != ""&& Request.QueryString["AuPieza12"] !=
""&& Request.QueryString["CalPieza18"] != "" && Request.QueryString["AuPieza18"] != ""&&
Request.QueryString["CalPieza30"] != ""&& Request.QueryString["AuPieza30"] != ""&&
Request.QueryString["CalPieza15"] != ""&& Request.QueryString["AuPieza15"] != ""&&
Request.QueryString["GranelKilo"] != ""&& Request.QueryString["Pieza180"] != "")
            {
                SqlCommand cmd = new SqlCommand("SP_IP_Capturar_Devoluciones", con);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@NoRemision",
int.Parse(Request.QueryString["NoRemision"].Trim()));
                cmd.Parameters.AddWithValue("@Folio",
Request.QueryString["Folio"].Trim());
                cmd.Parameters.AddWithValue("@12CalPieza",
int.Parse(Request.QueryString["CalPieza12"].Trim()));
                cmd.Parameters.AddWithValue("@12AuPieza",
int.Parse(Request.QueryString["AuPieza12"].Trim()));
                cmd.Parameters.AddWithValue("@18CalPieza",
int.Parse(Request.QueryString["CalPieza18"].Trim()));
                cmd.Parameters.AddWithValue("@18AuPieza",
int.Parse(Request.QueryString["AuPieza18"].Trim()));
                cmd.Parameters.AddWithValue("@30CalPieza",
int.Parse(Request.QueryString["CalPieza30"].Trim()));
                cmd.Parameters.AddWithValue("@30AuPieza",
int.Parse(Request.QueryString["AuPieza30"].Trim()));
                cmd.Parameters.AddWithValue("@15CalPieza",
int.Parse(Request.QueryString["CalPieza15"].Trim()));
                cmd.Parameters.AddWithValue("@15AuPieza",
int.Parse(Request.QueryString["AuPieza15"].Trim()));
                cmd.Parameters.AddWithValue("@GranielKili",
double.Parse(Request.QueryString["GranelKilo"].Trim()));
                cmd.Parameters.AddWithValue("@180Pieza",
int.Parse(Request.QueryString["Pieza180"].Trim()));
                con.Open();
                cmd.ExecuteNonQuery();
                con.Close();
                Response.Write("Agregado");
            }
            else
            {
                Response.Write("No Agregado");
            }
        }
        catch
        {
            Response.Write("Sin Datos");
        }
```

Permisos *Android Manifest:*

- Agregar permisos para que la aplicación tenga acceso a internet
- Agregar permisos para la localización de GPS (sistema de posicionamiento global)
- Agregar permisos para consultar el estado del teléfono y tener acceso al *IMEI*

- Agregar permisos para permitir que la aplicación guarde archivos en el almacenamiento externo

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

MainActivity.java

Con la asignación de los permisos en Android Manifestó se prosigue a crear el código para obtener el Imei del teléfono.

```
public void ObtenerImei(){
    try {
        manager= (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
        StringBuilder builder = new StringBuilder();
        builder.append(manager.getDeviceId());
        IMEItelefono.setText(builder.toString());
    }catch (Exception e){
        Toast.makeText(getApplicationContext(),"Debes Aceptar los
permisos",Toast.LENGTH_SHORT).show();
    }
}
```

Para iniciar sesión en la aplicación se procede a asignar el nombre y apellido del operador almacenándolos en las variables NombreUsuario y ApellidoUsuario, y se hace una validación en donde se revisa que los campos no estén vacíos, en el caso de estar vacíos se despliega un mensaje de alerta es necesario llenar este campo, pasando la validación se asignan los valores a una cadena *string* que se envía al método EnviarLogin.

```
public void login(View view){
    if
(TextUtils.isEmpty(NombreUsuario.getText().toString())||TextUtils.isEmpty(Apellli
doUsuario.getText().toString())){
        NombreUsuario.setError("Es Necesario Llenar Este Campo");
        NombreUsuario.requestFocus();
        ApelllidoUsuario.setError("Es Necesaario Llenar Este Campo");
        ApelllidoUsuario.requestFocus();
    }else {
        String cadenalogin = "http://"+ippublica+"/AndroidApp/Login.aspx?Nombre="
+NombreUsuario.getText().toString().replace(" ","_") + "&AP=" +
ApelllidoUsuario.getText().toString();
        EnviarLogin(cadenalogin);
        guardarpreferencias();
    }
}
```

Se hace una consulta a la base de datos con los datos del usuario y nos regresa la información correspondiente al usuario, se procede a validar la información capturada por el usuario y la que se ha consultado a la base de datos, en caso de

cumplir con todas las condiciones se pasa a la siguiente actividad con la siguiente linea de código. *Intent intent = new Intent(getApplicationContext(),Menu.class);*. En caso contrario de no estar registrado se procede a lanzar el menú de registro de usuario con el método dialogoimei(), en caso de no cumplir con ninguna condición nos lanzara un mensaje donde nos indica que el usuario no existe.

```
public void EnviarLogin(String URL){
    Log.d("url: ",URL);
    RequestQueue queue = Volley.newRequestQueue(this);
    StringRequest stringRequest =  new StringRequest(Request.Method.GET, URL, new
Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            try{
                if (response.length()>0){
                    ja = new JSONArray(response);
                     idpersona = ja.getString(0);
                    String nombre = ja.getString(1);
                    String apellidop=ja.getString(2).replace(" ","");
                    String imei = ja.getString(3).replace(" ","");

                    if (imei.equals(IMEItelefono.getText().toString())){
if(nombre.equals(NombreUsuario.getText().toString())&apellidop.equals(ApelllidoUs
uario.getText().toString()) ){
                            Intent intent = new
Intent(getApplicationContext(),Menu.class);
                            intent.putExtra("idpersona",idpersona);
                            intent.putExtra("nombre",nombre);
                            intent.putExtra("apellido",apellidop);
                            intent.putExtra("imei",imei);
                            finish();
                            startActivity(intent);
                        }
                    }else if (imei.equals(cero)){
                            dialogoimei();
                    }else {
                        Toast.makeText(getApplicationContext(),"Este No Es Tu
Dispositivo",Toast.LENGTH_SHORT).show();
                    }
                }else {
                    Toast.makeText(getApplicationContext(),"El usuario no
existe",Toast.LENGTH_SHORT).show();
                }
            }
            catch (JSONException e){
                e.printStackTrace();
                Log.i("error: ", e.getMessage());

Toast.makeText(getApplicationContext(),"error",Toast.LENGTH_LONG).show();
            }
        }
    },
            new Response.ErrorListener(){
                @Override
                public void onErrorResponse(VolleyError error) {

Toast.makeText(getApplicationContext(),error.toString(),Toast.LENGTH_LONG).show()
;
```

```
                    error.printStackTrace();
                }
            });

    queue.add(stringRequest);
}
```

ventana emergente que se lanza en el caso de que el operador no este registrado, con el método dialogoregistrado() se procede a abrir una nueva ventana para finalizar el registro.

```
public void dialogoimei(){
    new AlertDialog.Builder(this)
            .setTitle("Bienvenido Al Sistema de Asistencia Shaddai")
            .setMessage("Para poder continuar es necesario que registre este
telefono a Su Usuario")
            .setNegativeButton("Cancelar", new DialogInterface.OnClickListener()
{
                @Override
                public void onClick(DialogInterface dialog, int which) {

                }
            })
            .setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {

                    dialogregistrado();
                }
            }).create().show();
}
```

el método dialogregistrado() lanza una ventana emergente para finalizar el registro del operador haciendo la inserción del imei del teléfono al usuario del operador

```
public void dialogregistrado(){
    new AlertDialog.Builder(this)
            .setTitle("Sistema de Registro")
            .setMessage("Se ha registrado su telefono")
            .setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Intent intent = new
Intent(getApplicationContext(),MainActivity.class);
                    startActivity(intent);
                    finish();
                    enviarimei();
                }
            }).create().show();
}
```

Menu.java

Asistencia(): con este método se crea un Intent que nos direcciona a la ventana de asistencia, enviando sus datos idpersona, nombre y apellido.

```
public void Asistencia(View view){
    Intent i = getIntent();
    String Idpersona = i.getStringExtra("idpersona");
```

```
    String Nombre = i.getStringExtra("nombre");
    String Apellido = i.getStringExtra("apellido");

    Intent intent = new Intent(getApplicationContext(),Asistencia.class);
    intent.putExtra("idpersona",Idpersona);
    intent.putExtra("nombre",Nombre);
    startActivity(intent);
}
```

Remisiones(): con este método se crea un Intent que direcciona a la ventana de remisiones, enviando los datos del operador idpersona para consultar sus remisiones a cargo.

```
public void Remisiones(View view){
    Intent i = getIntent();
    String Idpersona = i.getStringExtra("idpersona");
    Intent intent = new Intent(getApplicationContext(),Remisiones.class);
    intent.putExtra("idpersona",Idpersona);
    startActivity(intent);
}
```

permisogps(): método que al ejecutarse otorga permiso de ocupar la posición actual del gps del dispositivo móvil.

```
public void permisogps(){
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
{
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,}, 1000);
    } else {
        locationStart();
    }
}
```

timergps: método que inicia un timer que se inicia después de 15 segundos de iniciar la actividad menú, comienza a recibir la latitud y longitud de nuestro dispositivo móvil y la almacena en las variable latitud y longitud, tras pasar los 15 segundos hace una consulta a la base de datos pidiendo la última posición guardada del dispositivo, tras recibir la petición compara la latitud y longitud actual con la que se encuentra en la base de datos, en caso de ser igual se procede a actualizar la hora en la que se hizo la petición, en caso de no ser iguales se procede a hacer una conversión de ambas posiciones transformándolas a metros, una vez que se genera la conversión se comprara la distancia actual con la de la base de datos, en caso de que la distancia sea mayor a 20.5 metros se hace una inserción en la base de datos, en caso de ser menor se vuelve a actualizar la hora.

```
public void timergps(){
    TimerTask task = new TimerTask() {
```

```
        @Override
        public void run()
        {
            String
pedirultimacoordenada="http://"+ippublica+"/AndroidApp/UltimaUbicacion.aspx?IdPer
sona="+id.getText().toString();
            pedirultimacoordenada(pedirultimacoordenada);
            if
(latitud.getText().toString().equals(latituddelabase.getText().toString())&longit
ud.getText().toString().equals(longituddelabase.getText().toString())){
                Log.d("LLENADO: ","SON IGUALES ACTUALIZAR HORA");
                String
actualizarhora="http://"+ippublica+"/AndroidApp/ActualizarHora.aspx?Hora="+er.get
Text().toString().replace(" ","")+"&IdPersona="+id.getText().toString();

                actualizarhoraultimopunto(actualizarhora);
            }else{

                if
(latituddelabase.getText().toString().equals(mop)&longituddelabase.getText().toSt
ring().equals(mop)){

                }else {
                    double
latuno=((Double.parseDouble(latituddelabase.getText().toString())*Math.PI)/180);
                    double
lonuno=((Double.parseDouble(longituddelabase.getText().toString())*Math.PI)/180);

                    double
latdos=((Double.parseDouble(latitud.getText().toString())*Math.PI)/180);
                    double
londdos=((Double.parseDouble(longitud.getText().toString())*Math.PI)/180);

                    double
d=6378.137*Math.acos(Math.cos(latuno)*Math.cos(latdos)*Math.cos(londdos-
lonuno)+Math.sin(latuno)*Math.sin(latdos));

                    double Mmetros = d*1000;
                    //metros.setText(String.valueOf(Mmetros+" metros"));

                    Log.d("TAG METROS",String.valueOf(Mmetros));

                    if (Mmetros >=20.5 ){
                        pedirultimacoordenada(pedirultimacoordenada);
                       String
mandarcoordenadas="http://"+ippublica+"/AndroidApp/InsertarCoordenadas.aspx?IdPer
sona="+id.getText().toString()+"&lat="+latitud.getText().toString().replace("
","")+"&long="+longitud.getText().toString()+"&fecha="+fechadia.getText().toStrin
g()+"&Hora="+er.getText().toString().replace(" ","");

                        if
(mop.equals(latitud.getText().toString())&mop.equals(longitud.getText().toString(
))){
                          Log.d("LLENADO: ","VACIO");
                       }else {
                           actualizarposicion(mandarcoordenadas);
                             Log.d("TAG :","-----------MANDO----
"+er.getText().toString());}
                       Log.d("TAG :","--NO SON IGUALES----");
                    }else {
                        String
actualizarhora="http://"+ippublica+"/AndroidApp/ActualizarHora.aspx?Hora="+er.get
```

```
Text().toString().replace(" ","")+"&IdPersona="+id.getText().toString();


                        actualizarhoraultimopunto(actualizarhora);
                        Log.d("LLENADO: ","SON IGUALES ACTUALIZAR HORA");
                    }
                }

            }

        }
    };
    timer.schedule(task, 15000, 60000);
}
```

Asistencia.java

Actualizartablero(): al ejecutarse este método hace luna petición a la base de datos solicitando los datos de la asistencias del día actual del operador.

```
public void actualizartablero(View view){
    String tablerohorario=
"http://"+ippublica+"/AndroidApp/ConsultarHorario.aspx?Idpersona="+id.getText().t
oString();
    enviarhorario(tablerohorario);
}
```

cargar(): al ejecutarse este método manda una inserción de su asistencia a la base de datos con el id del operador.

```
public void cargar(View view){
    clicks++;
    if (clicks==1){
        String registro
="http://"+ippublica+"/AndroidApp/InsertarAsistencia.aspx?IdPersona="+id.getText(
).toString();
        enviarcoordenadas(registro);

        String tablerohorario=
"http://"+ippublica+"/AndroidApp/ConsultarHorario.aspx?Idpersona="+id.getText().t
oString();
        enviarhorario(tablerohorario);
    }else if (clicks==2){
        String tablerohorario=
"http://"+ippublica+"/AndroidApp/ConsultarHorario.aspx?Idpersona="+id.getText().t
oString();
        enviarhorario(tablerohorario);
    } else if (clicks==3){
        String tablerohorario=
"http://"+ippublica+"/AndroidApp/ConsultarHorario.aspx?Idpersona="+id.getText().t
oString();
        enviarhorario(tablerohorario);
        clicks=clicks*0;
        Toast.makeText(getApplicationContext(),"Ya Puedes Registrar Tu
Asistencia",Toast.LENGTH_SHORT).show();
    }
    else {
        Log.d("TAG: ",String.valueOf(clicks));
    }
}
```

Remisiones.java

dialogoremision() al ejecutarse este método lanza una ventana emergente para que el operador seleccione una de las dos opciones. En el caso de seleccionar devolución lanza un intent que nos manda a la ventana donde se procede a capturar las piezas que serán devueltas junto con su folio.

```
public void dialogoremision(){
    new AlertDialog.Builder(this)
            .setTitle("           Selecciona una opcion")
            .setNegativeButton("Entregado", new DialogInterface.OnClickListener()
{
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    entregaremision();
                }
            })
            .setPositiveButton("Devolucion", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {

                    Intent intent = new
Intent(getApplicationContext(),Devolucion.class);
                    intent.putExtra("folioremision",aux_folio);
                    startActivity(intent);
                }
            }).create().show();
}
```

entregaremision() al ejecutarse este método lanza una ventana emergente donde podremos ingresar el folio de la remisión que fue entregada.

```
public void entregaremision(){
    final AlertDialog.Builder mBuilder = new
AlertDialog.Builder(Remisiones.this);
    View mView = getLayoutInflater().inflate(R.layout.remisionentregaga,null);
    final EditText folioentrega
=(EditText)mView.findViewById(R.id.folioentregaremision);
    final TextView preview =(TextView)mView.findViewById(R.id.texto);
    preview.setText(aux_folio);
    mBuilder.setNegativeButton("Cancelar", new DialogInterface.OnClickListener()
{
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });
    mBuilder.setPositiveButton("Enviar", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {

        }
    });
    mBuilder.setView(mView);
    final AlertDialog dialog = mBuilder.create();
    dialog.show();
    dialog.getButton(AlertDialog.BUTTON_POSITIVE).setOnClickListener(new
View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            String cadenalogin =
"http://"+ippublica+"/AndroidApp/Devolucionsp.aspx?NoRemision="
+preview.getText().toString() + "&Folio=" + folioentrega.getText().toString();
            devolucionentregada(cadenalogin);

            String cadenaobs=
"http://"+ippublica+"/AndroidApp/Dobs.aspx?Remision="
+preview.getText().toString() + "&Obs=" +so.replace(" ","_");
            actualizardevobs(cadenaobs);

            dialog.dismiss();
        }
    });
}
```

actualizarremision() al ejecutarse este método hace una petición a la base de datos para actualizar los datos de la lista.

```
public void actualizarremision(View view){
    Intent i = getIntent();
    String Idpersona = i.getStringExtra("idpersona");

    String consulta
="http://"+ippublica+"/AndroidApp/Remisiones.aspx?IdPersona="+Idpersona;
    Log.d("Consulta: ",consulta);
    EnviarRecibirRutas(consulta);
}
```

confirmado() al ejecutarse este método lanza una ventana donde nos da la información sobre la inserción que se hizo al base de datos correspondiente a la entrega de la remisión.

```
public void confirmado(String confirmacion) {
    String alerta = confirmacion;

    if (alerta.equals(agregado)){
        new AlertDialog.Builder(this)
                .setTitle("Shaddai: Respuesta")
                .setMessage("Su Peticion ha sido "+alerta)
                .setPositiveButton("Aceptar", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        actualizarremision();
                    }
                }).create().show();
    }else if (alerta.equals(noagregado)){
        new AlertDialog.Builder(this)
                .setTitle("Shaddai: Respuesta")
                .setMessage("Su Peticion ha sido "+alerta+" Verifique sus datos")
                .setPositiveButton("Aceptar", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                    }
                }).create().show();
    }else if (alerta.equals(sindatos)){
        new AlertDialog.Builder(this)
                .setTitle("Shaddai: Respuesta")
```

```
                .setMessage("Verifique  que sis datos no esten vacios")
                .setPositiveButton("Aceptar", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                    }
                }).create().show();
    }
}
```

Devolucion.java

confirmacion(): al ejecutarse este método recibe los datos capturados en los campos que han sido habilitados, lanza una ventana emergente para confirmar si los datos que han sido capturados están correctos.

```
public void confirmacion(String f,String c12,String au12,String c18,String
au18,String c30,String au30,String c15,String au15,String gra, String c180){
   final String folio=f;
    final String Cal12=c12;
    final String aur12=au12;
    final String cal18=c18;
    final String aur18=au18;
    final String cal30=c30;
    final String aur30=au30;
    final String cal15=c15;
    final String aur15=au15;
    final String gran=gra;
    final String cal180=c180;
    new AlertDialog.Builder(this)
            .setTitle("Shaddai: Verifique si los datos son correctos")
            .setMessage("¿Esta Seguro que seguro de que los datos son
correctos?")
            .setNegativeButton("Cancelar", new DialogInterface.OnClickListener()
{
                @Override
                public void onClick(DialogInterface dialog, int which) {
                }
            })
            .setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    String cadenalogin =
"http://"+ippublica+"/AndroidApp/Devolucionp.aspx?NoRemision=" +folioremision +
"&Folio=" +
folio+"&CalPieza12="+Cal12+"&AuPieza12="+aur12+"&CalPieza18="+cal18+"&AuPieza18="
+aur18+"&CalPieza30="+cal30+"&AuPieza30="+aur30+"&CalPieza15="+cal15+"&AuPieza15=
"+aur15+"&GranelKilo="+gran+"&Pieza180="+cal180;
                    actualizardevolucion(cadenalogin);
                    String obs =obser.getText().toString();
                    String nobs= obs.replace(" ","_");
                    String cadenaNormalize = Normalizer.normalize(nobs,
Normalizer.Form.NFD);
                    String cadenaSinAcentos =
cadenaNormalize.replaceAll("[^\\p{ASCII}]", "");
                    String cadenaobs=
"http://"+ippublica+"/AndroidApp/Dobs.aspx?Remision=" +folioremision + "&Obs=" +
cadenaSinAcentos.replace(",","_");
                    actualizardevobs(cadenaobs);                }
            }).create().show();
}
```

5.5. Pruebas de la App sobre distintos dispositivos

Ya finalizada el desarrollo de la aplicación móvil, se procede a hacer una serie de pruebas en distintos dispositivos móviles. La finalidad de realizar este tipo de pruebas es para analizar el comportamiento de la app en diferentes versiones de Android y también si se necesitara requerimientos muy específicos al momento de utilizar un dispositivo móvil.

Especificaciones *Samsung Galaxy j7:*

- Procesador: *Exynos* 7580 *1.5GHz*
- *RAM*: *1.5GB*
- Pantalla: 5.5", 720 x 1280 *pixels*
- Versión de *Android*: *Android 5.1Lollipop*

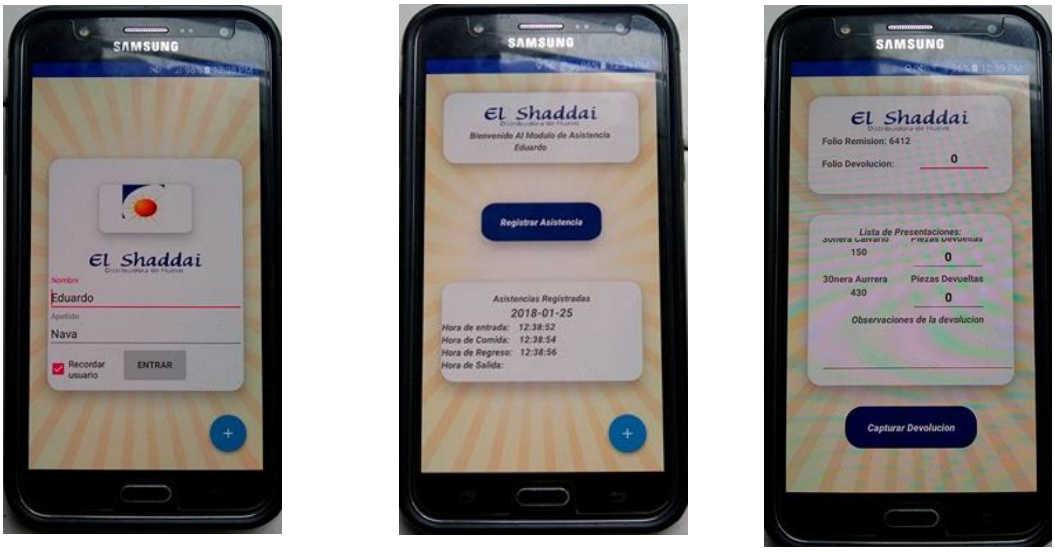


Figura 5.11: implementación Galaxy j7

Fuente: edición propia

No presento problemas con el diseño, todos sus elementos estuvieron en su lugar, reacciono de manera eficiente al activar el GPS (sistema de posicionamiento global) al igual que capturar la ubicación de este, este no presento las ventanas pidiendo que el usuario le diera permisos al usuario de usar características del dispositivo.

Especificaciones *Sony Xperia E5*:

- Procesador: *MediaTek 6755* 4 núcleos a *1,3GHZ.*
- *RAM: 1.5 GB.*

- Pantalla: 5 pulgadas con resolución *HD* (1.280 x 720 *píxeles*).
- Versión de *Android*: *Android 6.0 Marshmallow.*

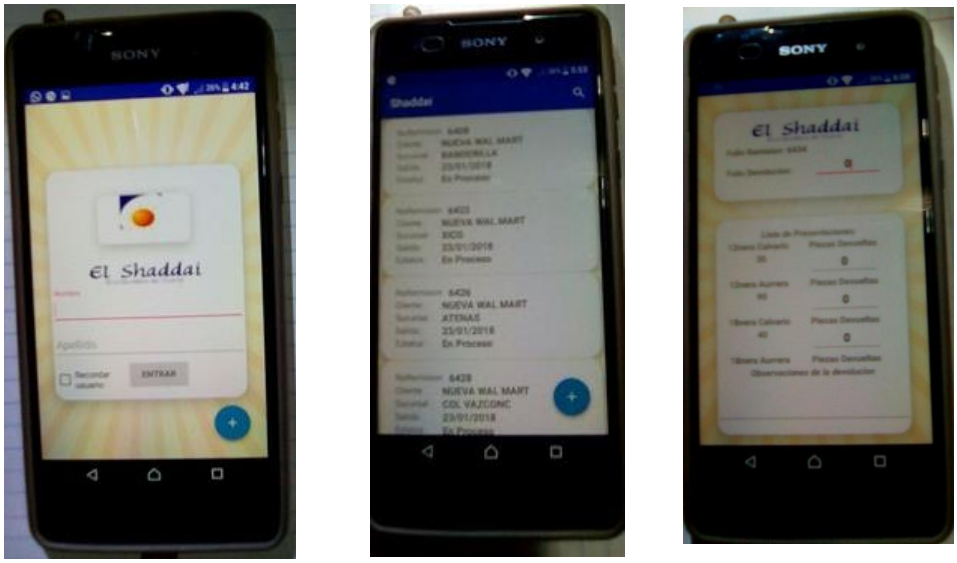


Figura 5.12: implementación Sony Xperia E5

Fuente: edición propia

No se presentaron problemas de diseño, todos los elementos permanecieron en su lugar correspondiente, presento problema al momento de iniciar el GPS (sistema de posicionamiento global) al igual un retraso al momento de capturar su ubicación GPS(sistema de posicionamiento global), se lanzaron las ventanas de permisos para utilizar ciertas características del dispositivo,

Especificaciones *Moto E4*:

- Procesador: *MediaTek* de cuatro núcleos de *1.3GHz*
- *RAM: 2 GB.*
- Pantalla: 5 pulgadas *HD (1280x720) 294 ppp, cristal 2.5D.*
- Versión de *Android: Android 7.1 Nougat.*

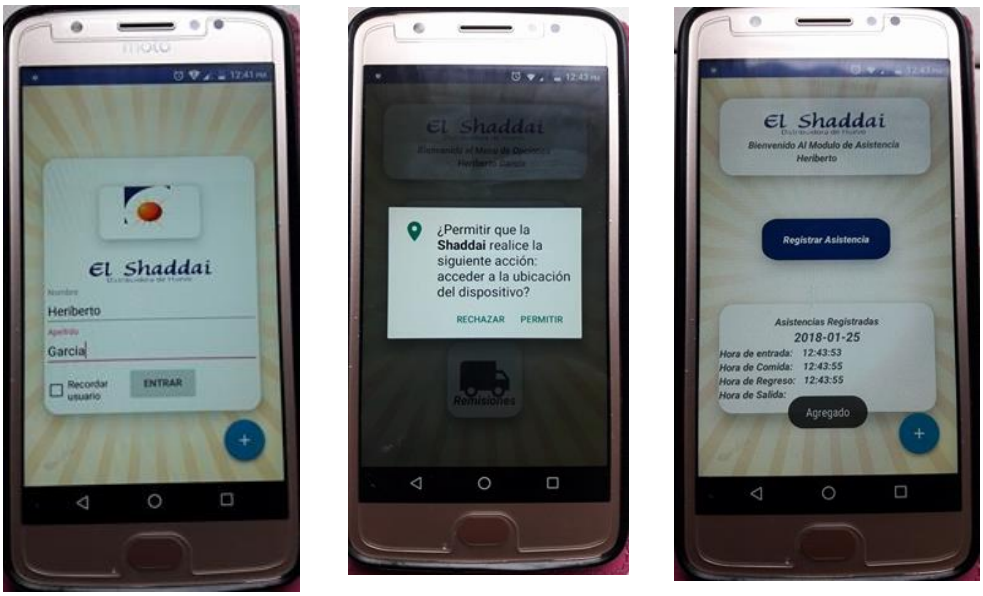


Figura 5.13: implementación Moto E4

Fuente: edición propia

No presento problemas al momento con el diseño, se detectaron detalles en reconocer el logo de la aplicación, la remplazo por un logo default de Android, presento problemas al usar el GPS (sistema de posicionamiento global), no se activó y no permitió él envió de la posición GPS (sistema de posicionamiento global) del dispositivo se lanzaron las ventanas de permisos para utilizar ciertas características del dispositivo.

5.6. Implementación de la App

Ya finalizada la fase de diseño y codificación procederemos a la fase de implementación, dentro de esta fase se visualizará las pantallas que ya fueron diseñadas anteriormente pero ya siendo implementadas dentro de un entorno real. Como primera fase de implementación procederemos a generar el registro de un chofer dentro de la aplicación. Para muestra de la funcionalidad de la Aplicación ocuparemos el chofer llamado Eduardo Nava para proceder a registrarnos y realizar las actividades correspondientes.

Como primer paso escribimos su nombre y su apellido dentro de los campos de texto, procedemos a dar en “Entrar” para comenzar nuestro registro.

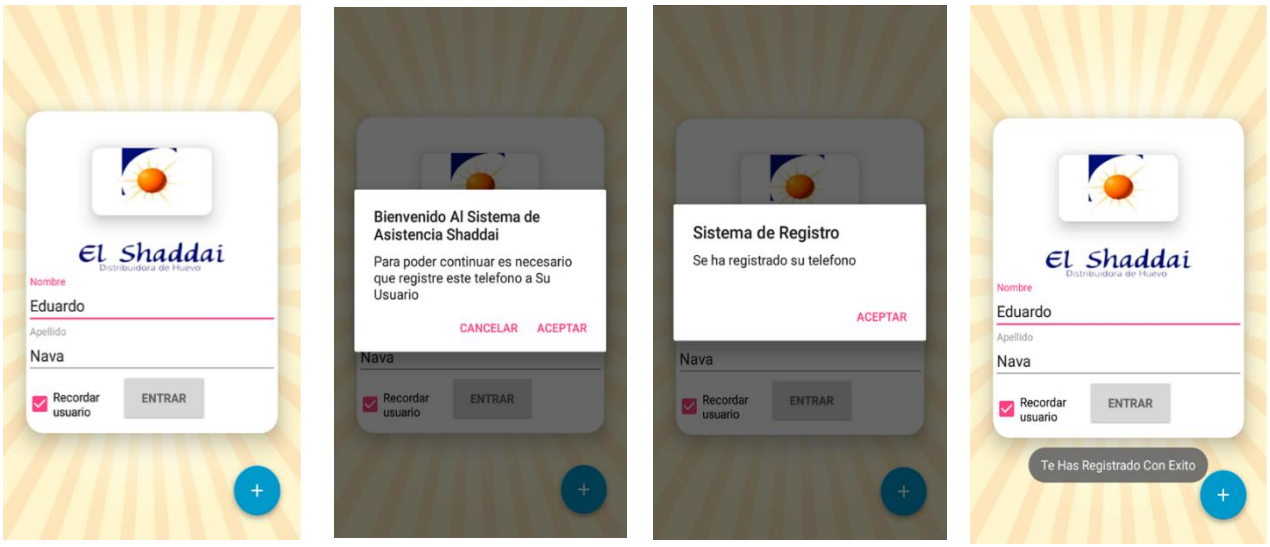


Figura 5.14: pruebas de implementación de registro

Fuente: edición propia

una vez que se finalizado nuestro proceso de registro podremos iniciar sesión presionando nuevamente en el botón Entrar, ya realizada esta acción tendremos acceso a una ventana como se muestra a continuación. Procederemos a presionar en la opción que dice asistencia, ya dentro de la ventana perteneciente al módulo de asistencias procederemos a registrar las primeras tres entradas y podremos observar que nos marca la hora, minutos y segundos en los que se realizó el chequeo.

Figura 5.15: pruebas de implementación menú y asistencia.

Fuente: edición propia

Tomando como referencia la ventana de menú que se muestra en el paso anterior, seleccionaremos la opción que dice remisiones, dentro de esta ventana podremos ver todas las remisiones que el chofer tiene asignadas a su cargo. Como se puede observar se da una breve información de las remisiones como es su cliente, sucursal, fecha de salida y el estatus de entrega en que se encuentra. Procederemos a realizar una devolución, para ello basta con seleccionar cualquiera de los elementos de la lista que se muestra, como vemos se abre una ventana emergente donde nos pide seleccionar una de las dos opciones disponibles, seleccionamos entregado, ahora nos mostrara una ventana donde podremos ingresar el folio de entregado, insertamos el folio y daremos en enviar, una vez enviado nos dará la respuesta por parte del servidor avisándonos que la petición fue aceptada y fue registrada la remisión con su folio de entregado.

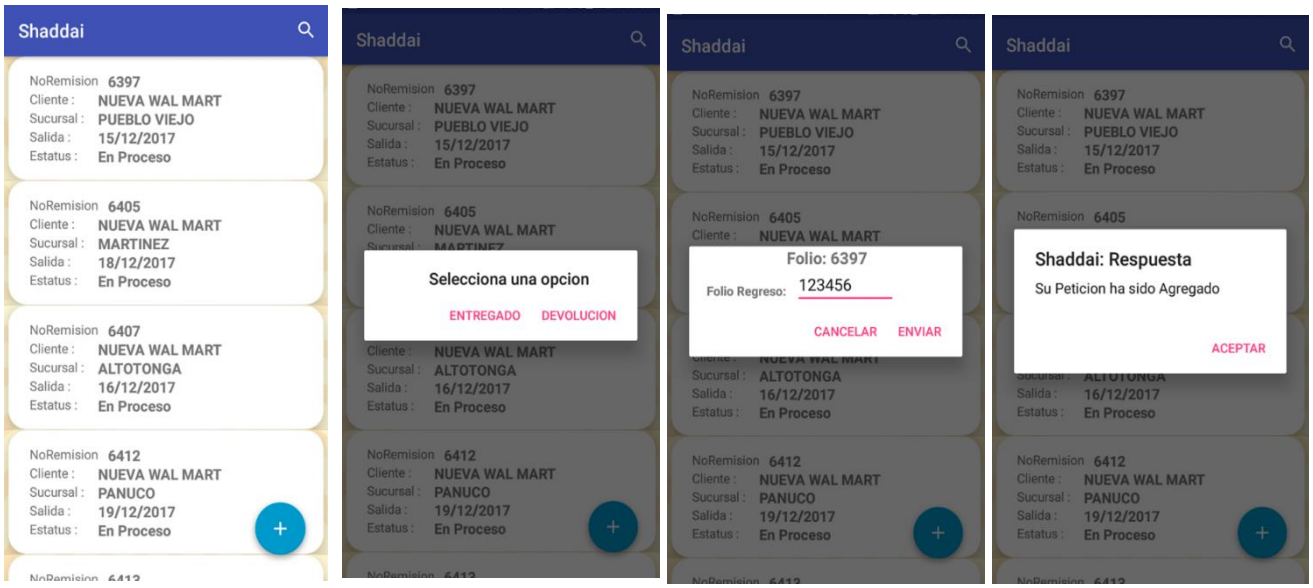


Figura 5.16: pruebas de implementación de entregas

Fuente: edición propia

Tomando como referencia la ventana emergente donde nos pide seleccionar la opción de entrega o devolución, procederemos a seleccionar la opción devolución, para esta ventana nos mostrar todas las presentaciones que contiene esta remisión asignada, dentro de esta ventana podremos ingresar el folio de devolución en la parte superior como se muestra en la siguiente imagen, como prueba de validación en la primer imagen se ingresaron devoluciones mayores a las cantidades de piezas en existencia, como podemos observar se pone una un mensaje de alerta declarando una acción invalida en cada una de las presentaciones.

Para proceder a insertar nuestra devolución se insertan cantidades acordes a las que tenemos en existencia actualmente, ingresamos algún tipo de observación en la parte inferior declarando el motivo de la devolución de cada una de las piezas, para este ejemplo solo dejaremos como sin observación y procedemos presionar el botón capturar devolución, a continuación nos mostrara una ventana de confirmación, daremos aceptar y finalmente tendremos un mensaje de que nuestra petición fue insertada con éxito.

Figura 5.17: pruebas de implementación de devolución

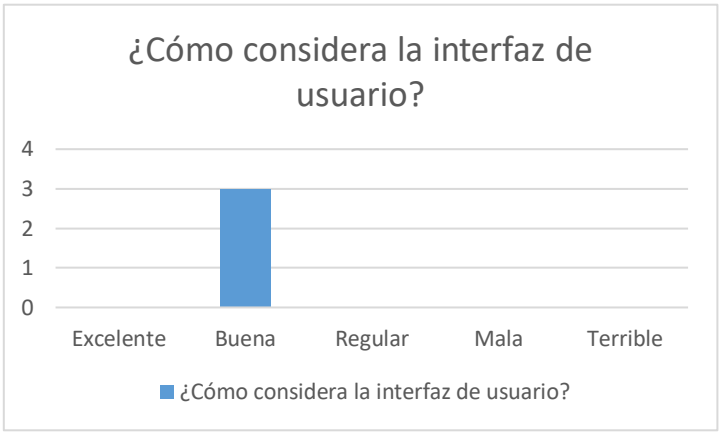
Fuente: edición propia

En conclusión Tras haber descrito las herramientas que se fueron empleando para el desarrollo de este proyecto, se concluye que, se puede crear aplicaciones móviles robustas capaces de recibir y enviar información e implementar el uso de sensores del dispositivo móvil en el que se encuentra alojada, con la implementación de una correcta metodología de desarrollo de software que sea flexible a los cambios inesperado en el transcurso de la creación del producto y que brinde herramientas de recolección de requerimientos fáciles de implementar nos ayuda a nosotros como desarrolladores a la creación de múltiples documentos extensos, ahorrándonos tiempo de documentación para enfocarse a la corrección de errores o cambios.

6. Resultados

6.1. Evaluación de eficiencia con administrador

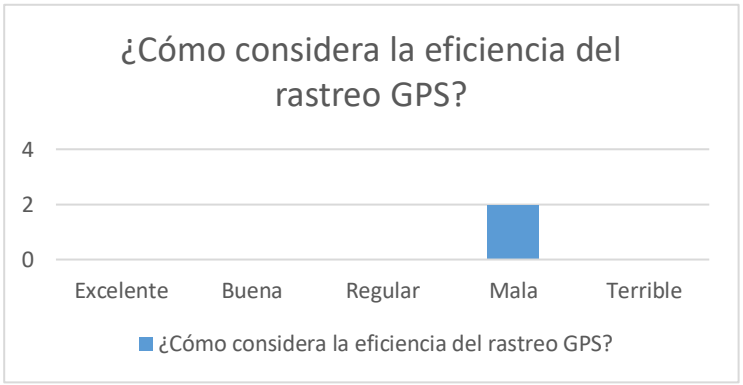
Con la finalidad de conocer la opinión de los administradores que usaron la aplicación web que fue anexada a su sistema, se procede a aplicar una encuesta para conocer sus preferencias referentes al entorno web y la recepción de la información correspondiente a la implementación de la aplicación móvil.



Cuadro 6.1: estadística pregunta administrador 1

Fuente: edición propia

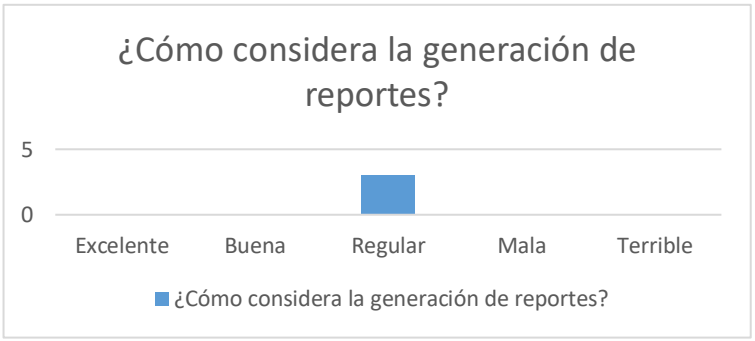
Se obtuvo que el 100% de los usuarios la considera buena, concluyendo que se obtuvo una interfaz intuitiva y fácil de manipular.



Cuadro 6.2: estadística pregunta administrador 2

Fuente: edición propia

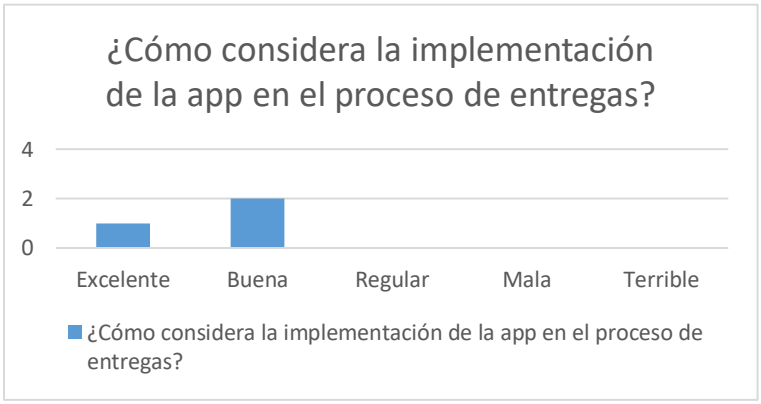
Se obtuvo que el 100% de los usuarios considera que la eficiencia del rastreo GPS es mala, concluyendo que la implementación del rastreo vehicular no cumplió con las expectativas esperadas.



Cuadro 6.3: estadística pregunta administrador 3

Fuente: edición propia

Se obtuvo que el 100% de los usuarios considera que la generación de los reportes es regular, concluyendo que formato empleado no logro las expectativas esperadas, esto debido al constante cambio de los formatos de reporte.



Cuadro 6.4: estadística pregunta administrador 4

Fuente: edición propia

Se obtuvo que el 10% de los usuarios considero que la integración de la app en el proceso de entrega es excelente y el 90% de los usuarios la consideran buena, concluyendo con un resultado satisfactorio referente a la mejora del proceso de entregas.

Anteriormente el proceso de las devoluciones se llevaba de manera manual, desde el momento en que las remisiones llegaban a los encargados de repartir las remisiones entre los operadores, hasta el momento de recibir la información de cada devolución. Para solventar esta problemática se optó por desarrollar un módulo dentro del sistema con el que cuenta la empresa.

Al ya existir un módulo donde se registran las remisiones que ya han sido cargadas por el personal administrativo se desarrolló la ventana de crear porteo que va

enlazada a la ventana de remisiones, es decir, dependiendo de las remisiones que han sido agregadas se podrá visual en una lista desplegable.

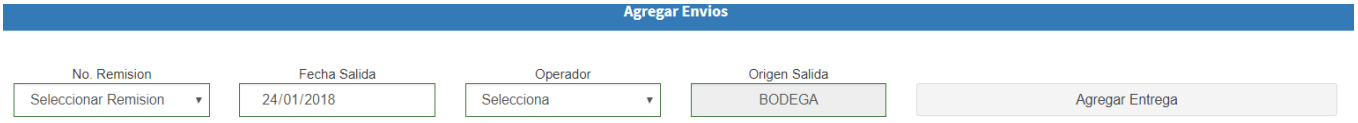


Figura 6.1: ventana *web* de registro de envíos

Fuente: edición propia

Como prueba de validación se procedió a intentar registrar una entrega sin seleccionar alguna remisión y algún operador en específico.

La prueba de validación resulto exitosa, no se debe de poder ingresar ninguna entrega si es que no se cuenta con una remisión asignada y un operador a cargo de su traslado.

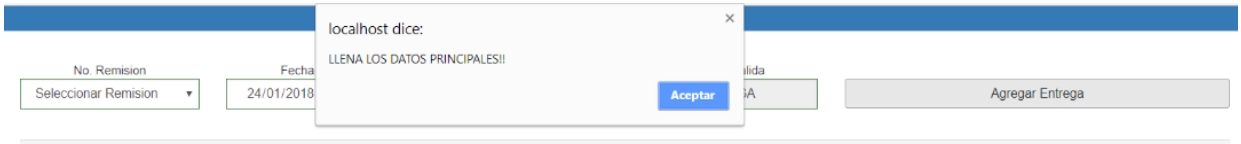


Figura 6.2: prueba de validación web

Fuente: edición propia

Para la recolección de la información que cada operador proporciona al encargado del mostrador, se genera una lista todos los días donde se anota cada una de las observaciones que se le dio al operador al momento de entregar su carga, este proceso ha sido uno de los más tardados ya que esta información ha tenido retrasos

hasta de 48 horas. Con la implementación de la aplicación móvil y la creación de la ventana de crear porteo se puede consultar esta información de una manera más rápida.

Con la creación de esta ventana se logró consultar información de una manera más rápida, al poder buscar las remisiones por operador, fecha de salida o por clientes logra agilizar el proceso de búsqueda de información como son las cajas que han sido entregadas y se logra obtener información de la causa de las devoluciones.

Figura 6.3: listado de envíos

Fuente: edición propia

para la generación de los reportes que se piden a diario ya no se necesita que los operadores lleguen a la bodega de la empresa para poder pasar sus observaciones al finalizar todas sus entregas, debido a que las observaciones ya se mandan a través de la aplicación móvil se puede generar este reporte de manera más rápida.

El Shaddai

Huevo El Calvario "EL SHADDAI"

Col. Melchor Ocampo, José de Emparan, N

Reporte de remisiones del dia

No Remision	Cajas Entregadas	Cliente	Sucursal	Operador	Salida	Observacion	Estatus
6397	16	NUEVA WAL MART	PUEBLO VIEJO	Eduardo	15/12/2017	Sin Observacion	Entregado
6405	20	NUEVA WAL MART	MARTINEZ	Eduardo	18/12/2017	sin observacion	Entregado
6407	13	NUEVA WAL MART	ALTOTONGA	Eduardo	16/12/2017	Sin Observacion	Entregado
6410	0	NUEVA WAL MART	ZAMORA CENTRO	Eduardo	16/12/2017	Sin Observacion	Entregado
6412	0	NUEVA WAL MART	PANUCO	Eduardo	19/12/2017	Sin Observacion	En Proceso
6413	0	NUEVA WAL MART	PEROTE	Eduardo	16/12/2017		En Proceso
6414	0	NUEVA WAL MART	TANTOYUCA	Eduardo	19/12/2017		En Proceso
6418	15	CHEDRAUI	TUXPAN CRYSTAL	Eduardo	10/05/2017	18ner calvario 200 piezas rotas 30nera calvario 400 piezas rotas	Entregado
6419	0	SORIANA	NARANJOS	Eduardo	15/12/2017		En Proceso
28869	2	CHEDRAUI	TUXPAN CRYSTAL	Eduardo	10/05/2017	Sin Observacion	Entregado
6423	0	NUEVA WAL MART	TUXPAN	Eduardo	01/01/2018	sin obervacion	En Proceso

Figura 6.4: reporte de envíos

Fuente: edición propia

Con la implementación del rastreo GPS (sistema de posicionamiento global) implementado en los dispositivos móviles de los operadores se logró obtener una visualización más exacta por donde es que los transportistas conducen, con esto se logró evaluar los tiempos de traslado, el coste de combustible que se ha gastado por cada una de las vías que han tomado, incluso se planifica nuevas rutas de envió con la finalidad de reducir estos tiempos y costos de gasolina

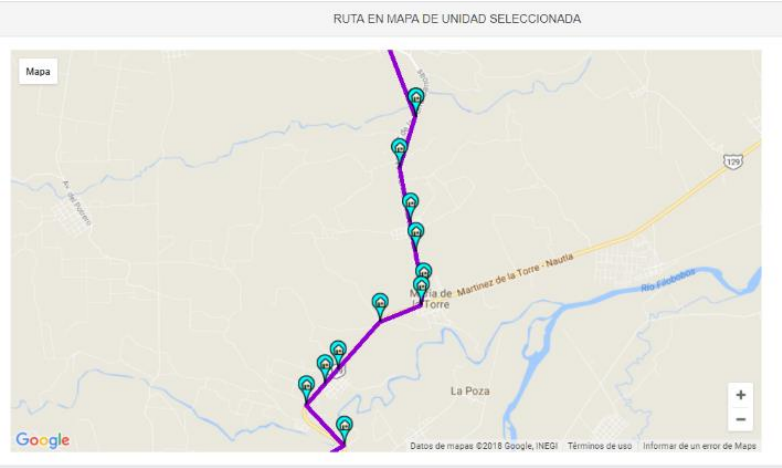


Figura 6.5: rastreo vehicular

Fuente: edición propia

Resultados obtenidos		
Actividades	Antes	Actualmente
Informes	5 a 10 horas	7 horas
Captura de devoluciones	24 a 48 horas	4 a 10 horas
Información actualizada	24 a 48 horas	4 a 10 horas
Inventario finalizado	48 horas	24 horas
Rastreo vehicular	Mensajes por WhatsApp mostrando su ubicación	Interfaz que muestra la ruta por la cual ha pasado el operador

Cuadro 6.5: resultados obtenidos por parte del administrador

Fuente: edición propia

6.2. Evaluación de eficiencia como cliente

Con la finalidad de conocer la opinión de los operadores que usaron la aplicación se procede a aplicar una encuesta para conocer sus preferencias referentes a la aplicación. Para ello se realizó un muestreo

El muestreo es el proceso de seleccionar un conjunto de individuos de una población con el fin de estudiarlos y poder caracterizar el total de la población. [49]

Para ello se estableció un margen de error del 5% y un nivel de confianza del 95% a un universo total de 10 personas.

Muestra

$$\boldsymbol{n = \frac{z^2 * p * q * N}{e^2(N-1) + Z^2 * P * Q}}$$

N=Población

n=Muestra

p=Probabilidad a favor

q=Probabilidad en contra

z=Nivel de confianza

e=Error de muestra

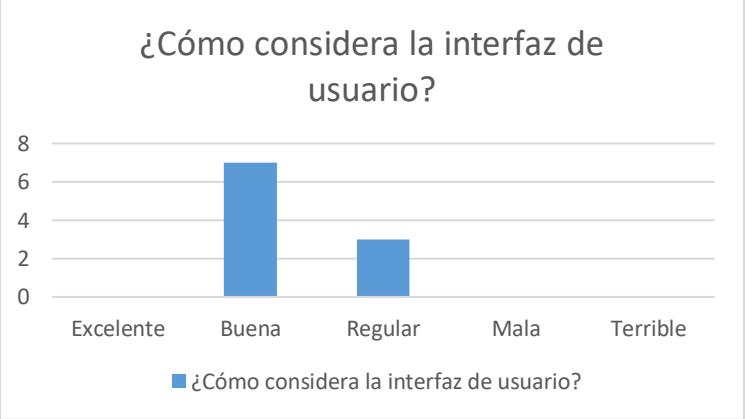
$$n = \frac{1.96^2 * 0.5 * 0.5 * 10}{0.05^2(10-1) + 1.96^2 * 0.5 * 0.5}$$

$$n = \frac{9.604}{0.9829}$$

$$n = 9.7710$$

$$n = 10$$

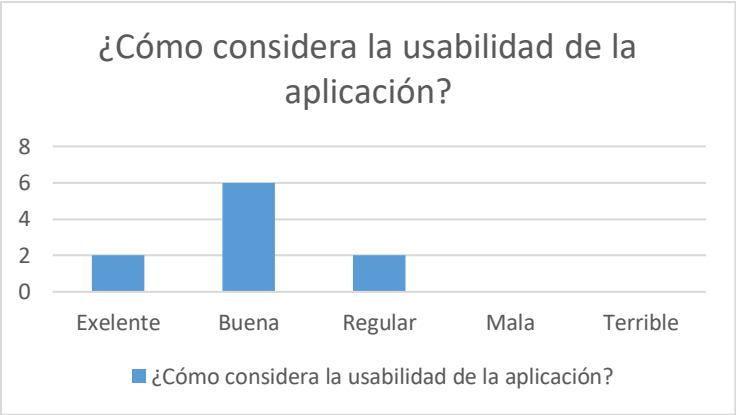
Se procedió a aplicar la encuesta a los 10 operadores que ocuparon la aplicación, done las 10 personas representan al 100% de los operadores. Aplicando la encuesta se obtuvieron los siguientes resultados.



Cuadro 6.6: estadística pregunta cliente 1

Fuente: edición propia

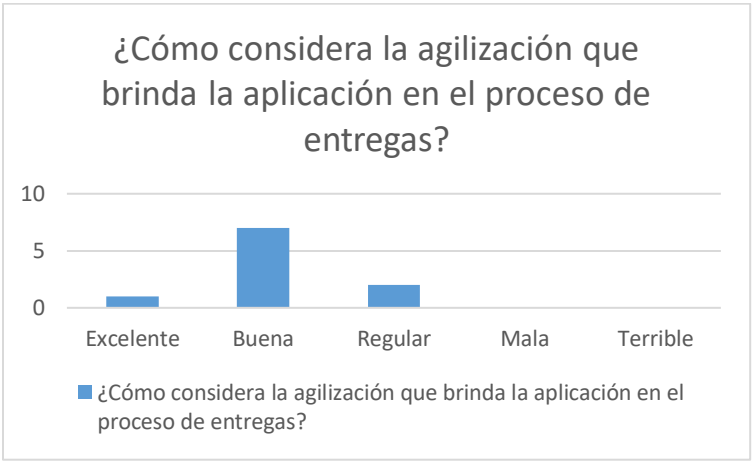
Se obtuvo que 70% de los operadores consideran que la interfaz de usuario es buena y 30% de los operados consideran que es regular, concluyendo que la interfaz diseñada es bien recibida por parte de los usuarios



Cuadro 6.7: estadística pregunta cliente 2

Fuente: edición propia

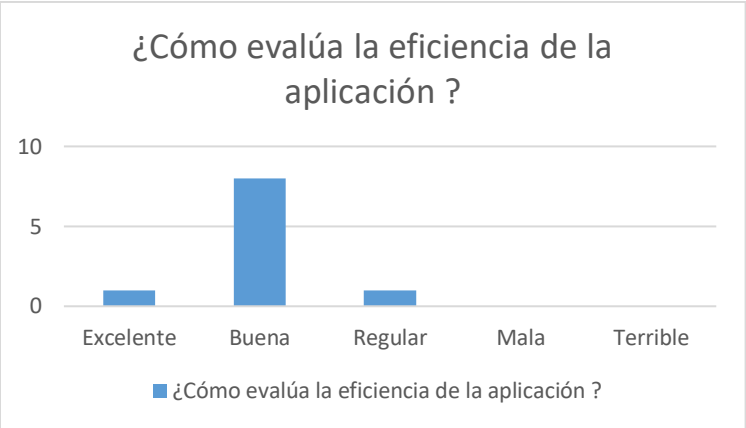
Se obtuvo que 20% de los operadores considera la usabilidad excelente, 60% operadores considera que es buena y finalmente 20% que es regular, concluyendo que la usabilidad de la aplicación es bien recibida por parte de los usuarios.



Cuadro 6.8: estadística pregunta cliente 3

Fuente: edición propia

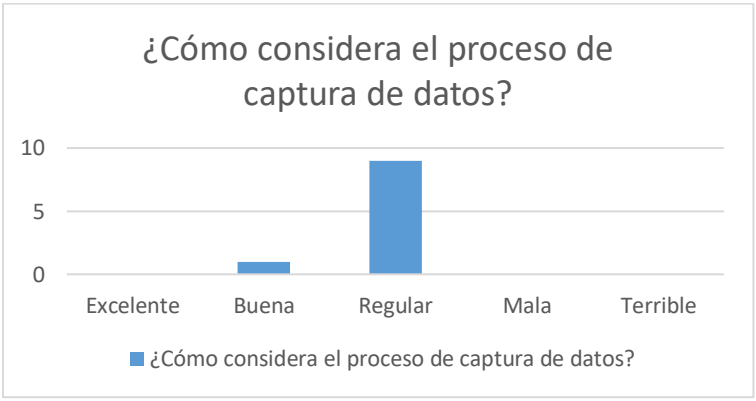
Se obtuvo que 10% de los operadores considera excelente la agilidad que proporciona, 70% de los operadores considera que es buena y finalmente 20% operadores considera que es regular, concluyendo que la aplicación móvil cumple con su objetivo de mejorar el proceso de entregas.



Cuadro 6.9: estadística pregunta cliente 4

Fuente: edición propia

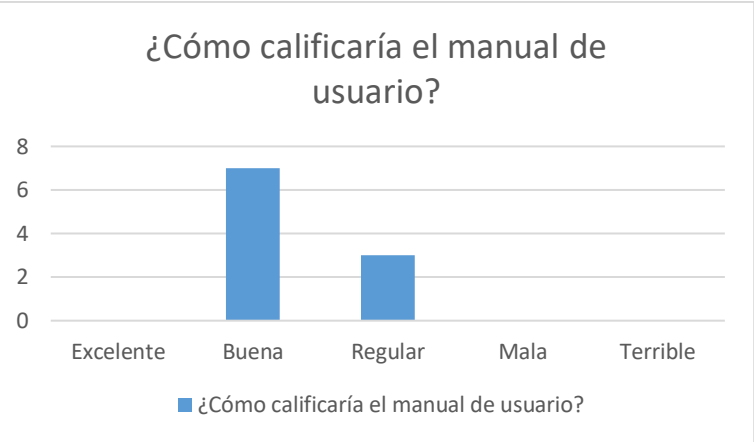
Se obtuvo que 10% de los operadores considera que la eficiencia de la aplicación es excelente, el 80% considera que es buena y el 10% considera que es regular, concluyendo que la aplicación cumple satisfactoriamente su función.



Cuadro 6.9: estadística pregunta cliente 5

Fuente: edición propia

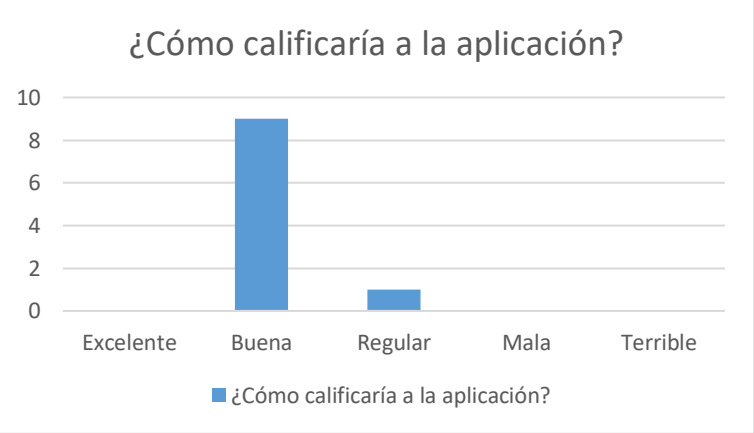
Se obtuvo que el 90% de los operadores considera que el proceso de captura de información es considerado regular, y el 10% lo considera buena, concluyendo que el proceso de captura es un poco laborioso en el caso de los operadores que no cuentan con mucha experiencia en el uso de aplicaciones móviles.



Cuadro 6.10: estadística pregunta cliente 6

Fuente: edición propia

Se obtuvo que el 70% de los operadores considera útil el uso del manual de usuario y el 30% lo considera regular, concluyendo un resultado positivo por aquellos usuarios que no cuentan con demasiada experiencia en el uso de aplicaciones.



Cuadro 6.11: estadística pregunta cliente 7

Fuente: edición propia

Se obtuvo que un 90% de los operadores considera buena la aplicación y un 10% la considera regular, se concluye satisfactoriamente que la aplicación es bien recibida por parte de los usuarios.

Anteriormente los operadores demoraban un lapso de 24 a 48 horas en poder entregar la información correspondiente a sus entregas, esto ocasionaba que la información no pudiera estar actualizada en tiempo y forma probando problemas

con el personal de la empresa el calvario cuando necesitaban consultar información relacionada a las entregas que fueron programadas.

Para solventar esta problemática se desarrolló una aplicación móvil para la plataforma Android, aprovechando que dentro del personal de la empresa la abundancia de este sistema operativo es mucho mayor al del sistema *IOS*. Se comenzó a diseñar ventas que fueran intuitivas para los operadores, se realizaron pruebas de validaciones donde se previene la ausencia de información en cada uno de los campos, así como evitar que se ingresaran cantidades de piezas devueltas a las que se tienen en existencia en el cargamento.

Figura 6.6: captura de devolución

Fuente: edición propia

Las pruebas de validación resultaron de manera exitosa, se evitó el ingreso de caracteres que no sean números y campos vacíos, la implementación de esta ventana para la captura de devoluciones resulto ser muy efectiva para el personal que labora en las oficinas de la empresa ya que se logró repartir el trabajo que aria una sola persona en un total de 10 personas.

Resultados obtenidos		
Actividades	Antes	Actualmente
Personal encargado de capturar devoluciones	1 persona	10 personas
Horas de captura	1:30 a 2 horas	1 hora
Información actualizada	24 a 48 horas	4 a 10 horas
Inventario finalizado	48 horas	24 horas

Cuadro 6.12: resultados obtenidos por parte del cliente

Fuente: edición propia

Como conclusión la implementación de la aplicación móvil se logró obtener resultados favorables en cuanto a tiempos de entrega de la información de cada una de las entregas que se realizaron en el trascurso del día, con el apoyo de un entorno web que apoya con la consulta de esta información, así como la creación de su correspondiente reporte se logró eliminar los lapsos de espera que en su momento fueron un gran problema. Con la implementación del rastreo GPS se logró mejorar la seguridad de los vehículos de reparto, con esto se evitó el uso mal intencionado de los vehículos y el desvió de las rutas establecidas.

7. Conclusión

Actualmente muchas empresas que están en constante crecimiento buscan la manera más eficiente de hacer su trabajo, ofrecer sus servicios, reducir tiempo y costos, para lograr sus objetivos buscan tecnologías novedosas que les ofrezcan soluciones optimas que los ayuden a lograr su cometido.

Con el constante desarrollo de las tecnologías móviles se puede contar con las herramientas necesarias que apoyaran para poder desarrollar aplicaciones móviles que nos ayuden a darle solución a los problemas de la empresa y a su vez hacer más eficiente su flujo de trabajo. Es por ello que, con el desarrollo de esta aplicación móvil, se puede comprobar que, con el uso de estas tecnologías podemos crear aplicaciones móviles que ofrezcan soluciones optimas a empresas que utilicen bases de datos y sus servicios dependan de esta.

Con el desarrollo de la presente tesis, la primera hipótesis que se planteó al iniciar esta investigación en que con el desarrollo de un sistema web para llevar el control de los usuarios que se han registrado en la aplicación, ha sido aceptada. Con los resultados obtenidos se logró resetear el estado de cada uno de los dispositivos para que el próximo operador pueda disponer del dispositivo.

La segunda hipótesis que se planteó en que consiste en el desarrollo de una aplicación móvil para capturar piezas devueltas y folios de remisiones, además de recudir tiempo, ha sido aceptada, con los resultados que se obtuvieron se logró reducir tiempos de captura de remisiones y folios.

La tercera hipótesis que se planteó en que consiste en la implementación de una aplicación móvil el operador lograra hacer más eficiente el proceso de entregas, ha sido aceptada, con los resultados que se lograron obtener se verifico que al interactuar el operador con el proceso de entregas se obtuvo mejores tiempos de entrega de información y a su vez contribuyo en el proceso de mantener la información actualizada en tiempo real.

La cuarta hipótesis que se planteó que consiste en la utilización de una aplicación móvil como medio de rastreo vehicular el personal de la empresa El Shaddai lograra

conocer la ubicación actual de cada vehículo de reparto. Ha sido rechazada, con los resultados obtenidos se logró visualizar el recorrido que genero el vehículo, pero a su vez el dispositivo móvil llego a perder la señal de la red móvil ocasionando que se perdiera el rastreo por lapsos de tiempo, otro inconveniente es el consumo de saldo del cual dispondría cada Smartphone, ocasionando un gran gasto monetario por parte de la empresa.

Los objetivos que se mencionaron al inicio de la presente tesis se cumplieron, con la utilización del GPS (sistema de posicionamiento global) del dispositivo y utilizando una interfaz intuitiva, los operadores podrán capturar la información que se les proporciona en el trascurso de sus entregas y posteriormente enviarla al personal encargado del área de porteo, con esto se logra agilizar su trabajo, reducir tiempo y hacer más eficiente el sistema ERP con el que cuentan.

7.1. Recomendaciones

- Capacitación de personal.
- Contar con dispositivos con sistema operativo Android.
- Aceptar todos los permisos que la aplicación solicite.
- Mantener el servidor en funcionamiento en todo momento.
- Mantener el dispositivo encendido en todo momento para no perder el rastreo GPS (sistema de posicionamiento global).
- Mantener actualizado Android Studio y proceder a actualizar la app conforme nuevas herramientas se vayan implementando para mejorar su funcionamiento.

7.2. Trabajos futuros

Para el desarrollo de esta aplicación se contempló utilizar el sistema operativo Android, sin embargo, a pesar de que la gran mayoría de los operadores de la empresa cuentan con un Smartphone que cuenta con esta plataforma algunos optan

por osar otro sistema operativo, sin embargo, con la metodología empleada en este trabajo, se puede realizar el desarrollo para otros sistemas operativos como *IOS*.

Con la culminación de este trabajo, se listan las áreas en donde la aplicación puede ser mejorada, con la finalidad de ser más eficiente al usuario y la empresa:

- Inicio de sesión: implementar un control de registro mediante correo electrónico Gmail y permitir mantener abierta la sesión al abrir la aplicación.
- Módulo de ventas: implementar un módulo para llevar un mejor registro de las ventas realizadas para los operadores encargados de distribuir producto a pequeños locales.
- Permitir seleccionar fotos de la galería de imágenes, la foto que se seleccione de las remisiones sea enviada al servidor de la empresa.

El presente trabajo solo se enfocó en el lado de los operadores dedicados a la entrega de porteo, con este mismo proyecto, se puede crear la aplicación, del lado de los operadores encargado de rutas y los clientes a los que se les provee producto.

Bibliografía

[1] A. S. Tanenbaum, Sistemas Operativos Modernos(3ra ed.), Mexico: PEARSON EDUCACIÓN, 2009.

[2] W. Stallings, Sistemas Operativos(5ta ed.), Madrid: Peason Education,, 2005.

[3] P. Aparicio, «Google lanza herramienta para portar aplicaciones de Android a IOS,» Actualidad Iphone, 6 febrero 2016. [En línea]. Available: https://www.actualidadiphone.com/google-lanza-herramienta-para-portar-aplicaciones-de-ios-a-android/. [Último acceso: 4 noviembre 2017].

[4] SiliconIndia, «From iOS 1 to iOS 9: How apple's iPhone OS has evolved,» SiliconIndia, 1 Febrero 2016. [En línea]. Available: https://news.siliconindia.com/technology/From-iOS-1-to-iOS-9-How-Apples-iPhone-OS-has-Evolved-nid-192104-cid-2.html/. [Último acceso: 20 octubre 2017].

[5] iPhoneros, «iPhoneros,» 2007. [En línea]. Available: https://iphoneros.com/wp-content/uploads/2016/06/logodeios10.jpg. [Último acceso: 12 diciembre 2017].

[6] M. Gargenta y M. Nakamura, Learning Android, United States of America: O'Reilly Media, 2014.

[7] . M. López Michelone, «La historia de Android,» 23 Septiembre 2013. [En línea]. Available: https://www.unocero.com/noticias/gadgets/smartphones/android/la-historia-de-android/. [Último acceso: 27 agosto 2017].

[8] J. Antonio, «Androides del pasado, recordando lo que una vez triunfó: HTC Dream,» 19 Julio 2013. [En línea]. Available: https://andro4all.com/2013/07/androides-del-pasado-htc-dream. [Último acceso: 22 noviembre 2017].

[9] Developer Android, «Arquitectura de la plataforma,» [En línea]. Available: https://developer.android.com/guide/platform/index.html. [Último acceso: 22 noviembre 2017].

[10] S. Pérochon y S. Hébuterne, Android guia de desarrollo de aplicaciones para Smartphone y Tabletas, eni.

[11] Android, «Historia de android,» Android, [En línea]. Available: https://www.android.com/intl/es-419_mx/history/#/icecream. [Último acceso: 23 octubre 2017].

[12] AndroidCurso, «Arquitectura de Android,» AndroidCurso, 13 Abril 2014. [En línea]. Available: http://www.androidcurso.com/index.php/99. [Último acceso: 23 octubre 2017].

[13] Á. J. Vico, «Arquitectura Android,» 17 Febrero 2011. [En línea]. Available: https://columna80.wordpress.com/2011/02/17/arquitectura-de-android/. [Último acceso: 11 octubre 2017].

[14] Software de Comunicacion, «Arquitectura Android,» Software de comunicacion, [En línea]. Available: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android. [Último acceso: 15 septiembre 2017].

[15] wikiwand, «Android cupcake,» [En línea]. Available: http://www.wikiwand.com/tr/Android_Cupcake. [Último acceso: 25 noviembre 2017].

[16] Google Sites, «Android 1.6 donut,» [En línea]. Available: https://sites.google.com/site/operatinsys/android-1-6-donut. [Último acceso: 13 noviembre 2017].

[17] J. Middleton, «telecoms,» 16 noviembre 2019. [En línea]. Available: http://telecoms.com/16312/android-serves-up-eclair/. [Último acceso: 24 noviembre 2017].

[18] D. Courbanou, «A Week With Google Android 2.2: Froyo,» 14 junio 2010. [En línea]. Available: http://www.channelfutures.com/best-practices/week-google-android-22-froyo. [Último acceso: 26 noviembre 2017].

[19] L. D. «Future Play Store apps will no longer run on Android 2.3 Gingerbread,» 21 noviembre 2016. [En línea]. Available: https://www.phonearena.com/news/Future-Play-Store-apps-will-no-longer-run-on-Android-2.3-Gingerbread_id88079. [Último acceso: 27 noviembre 2017].

[20] Androide central, «Android 3.x- Honeycomb,» 11 agosto 2016. [En línea]. Available: https://www.androidcentral.com/honeycomb. [Último acceso: 26 noviembre 2017].

[21] A. Castillo, «La historia de android: todas sus versiones,» 24 septiembre 2015. [En línea]. Available: http://www.poderpda.com/editorial/la-historia-de-android-todas-sus-versiones/. [Último acceso: 25 agosto 2017].

[22] Mi movil Android, «Android 4.1 Jelly Bean,» [En línea]. Available: http://mimovilandroid.com/tutoriales/android-4-1-jelly-bean/. [Último acceso: 26 noviembre 2017].

[23] J. Causey, «Google releases Android 5.0 Lollipop SDK and other tools,» 17 octubre 2014. [En línea]. Available: http://www.talkandroid.com/223473-google-releases-android-5-0-lollipop-sdk-and-other-tools/. [Último acceso: 26 noviembre 2017].

[24] J. Herrick, «Google names Marshmallow as the official treat for Android 6.0,» 17 agosto 2015. [En línea]. Available: http://www.talkandroid.com/261366-google-names-marshmallow-as-the-official-treat-for-android-6-0/. [Último acceso: 27 noviembre 2017].

[25] Android, «Android nougat,» [En línea]. Available: https://www.android.com/versions/nougat-7-0/. [Último acceso: 27 noviembre 2017].

[26] Android, «Android oreo,» [En línea]. Available: https://www.android.com/versions/oreo-8-0/. [Último acceso: 27 noviembre 2017].

[27] J. M. Acuña Morgado, «Características y tabla comparativa de los sistemas operativos móviles más usados,» 17 Marzo 2017. [En línea]. Available: http://jmacuna.tecnoblog.guru/2017/03/sistemas-operativos-moviles.html. [Último acceso: 15 noviembre 2017].

[28] Developer Android, «Paneles de control,» [En línea]. Available: https://developer.android.com/about/dashboards/index.htm. [Último acceso: 27 noviembre 2017].

[29] Eclipse, «Eclipse,» [En línea]. Available: https://www.eclipse.org. [Último acceso: 29 noviembre 2017].

[30] Developer Android, «Android Studio,» [En línea]. Available: https://developer.android.com/studio/index.html. [Último acceso: 1 diciembre 2017].

[31] Oracle, «Java SE Development Kit 8 Downloads,» [En línea]. Available: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. [Último acceso: 1 diciembre 2017].

[32] Developer Android, «Conoce Android Studio,» [En línea]. Available: https://developer.android.com/studio/intro/index.html?hl=es-419. [Último acceso: 5 diciembre 2017].

[33] L. N. Medina Velandia y W. . M. López López, «Escoger una metofología para desarrollar,» *Educacion en Ingenieria,* pp. 99-101, 2015.

[34] R. S. Pressman, Ingeniería del software. Un enfoque práctico, México, D.F:: McGRAW-HILL, 7 ed.

[35] S. M. Meléndez Valladarez, M. E. Gaitan y N. N. Pérez Reyes, «Metodologia ágil de desarrollo de software programacion extrema.,» Nicaragua, 2016.

[36] J. Gonza Chavez , «Introducción a la metodología XP,» 1 agosto 2010. [En línea]. Available: http://jgcprogramacion.blogspot.mx/2010/08/test.html. [Último acceso: 12 septiembre 2017].

[37] D. Bustamante y J. C. Rodriguez, «Metodologia-XP,» 12 Marzo 2014. [En línea]. Available: http://blogs.unellez.edu.ve/dsilva/files/2014/07/Metodologia-XP.pdf. [Último acceso: 25 octubre 2017].

[38] SQLite, «Acerca de SQLite,» SQLite, 9 Mayo 2000. [En línea]. Available: https://sqlite.org/about.html. [Último acceso: 12 octubre 2017].

[39] F. Rómmel, «SQLite: la base de datos embebida,» 12 mayo 2007. [En línea]. Available: https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida. [Último acceso: 4 octubre 2017].

[40] SRS, «Beneficios de un sistema de rastreo satelital,» SRS, [En línea]. Available: http://sistemasderastreosatelital.com/beneficios-de-un-sistema-de-rastreo-por-gps/. [Último acceso: 4 diciembre 2017].

[41] UTS Sistemas, «Sistemas de rastreo movil GPS,» UTS Sistemas, [En línea]. Available: https://www.utssistemas.com.mx/Apps-Moviles.html. [Último acceso: 20 diciembre 2017].

[42] fleetmatics, «Gestion de flotillas GPS,» fleetmatics, [En línea]. Available: https://www.fleetmatics.mx/empresarial/funcionalidad/movil-rastreo-app. [Último acceso: 20 diciembre 2017].

[43] GPSMonitor, «Rastreo GPS vehícular,» [En línea]. Available: http://gpsmonitor.com.mx/rastreo-gps-vehicular/. [Último acceso: 20 diciembre 2017].

[44] V. Gómez, «Diagrama de casos de uso,» 25 Abril 2015. [En línea]. Available: https://instintobinario.com/diagrama-de-casos-de-uso/. [Último acceso: 10 agosto 2017].

[45] R. Quintana Zabala, «Base de datos y su importancia dentro de una organizacion,» 3 Marzo 2017. [En línea]. Available: https://www.gestiopolis.com/bases-datos-importancia-dentro-una-organizacion/.

[46] C. amaya cordoba, «Normalizacion,» 12 Septiembre 2009. [En línea]. Available: http://vainilla15.blogspot.mx/2009/09/normalizacion.html. [Último acceso: 13 noviembre 2017].

[47] My SQL hispano, «Normalización de bases de datos,» 29 mayo 2003. [En línea]. Available: http://www.eet2mdp.edu.ar/alumnos/MATERIAL/MATERIAL/info/infonorma.pdf. [Último acceso: 28 diciembre 2017].

[48] V. L. Nieto Quintanilla, «Normalizacion,» 27 Mayo 2017. [En línea]. Available: https://vlnieto.wikispaces.com/file/view/Unidad+3+-+Normalizacion.pdf. [Último acceso: 1 septiembre 2017].

[49] C. Ochoa, «El muestreo: qué es y por qué funciona,» netquest, 15 Febrero 2015. [En línea]. Available: https://www.netquest.com/blog/es/blog/es/muestreo-que-es-porque-funciona. [Último acceso: 12 Noviembre 2017].

[50] J. T. Gironés, El Gran Libro De Android, Mexico: Marcombo, 2014.

[51] S. M. Ross Mistry, Introducing Microsoft SQL server 2014, Washington: Microsoft Press, 2014.

[52] Microsoft, «Fundamentos de la normalización de bases de datos,» Microsoft, 29 Junio 2017. [En línea]. Available: https://support.microsoft.com/es-bo/help/283878/description-of-the-database-normalization-basics. [Último acceso: 20 diciembre 2017].

[53] Norpic, «Todas las versiones del sistema operativo iOS de Apple,» Norpic, 14 Octubre 2017. [En línea]. Available: https://norfipc.com/celulares/todas-versiones-sistema-operativo-ios-apple.php. [Último acceso: 20 Noviembre 2017].

Anexos

A.1 Requerimientos del dispositivo móvil.

A continuación, se lista las características con las que debe contar el dispositivo móvil para la posterior instalación de la aplicación:

- Smartphone con la versión de *Android 5.0 Lollipop* o superior.
- Contar con GPS (sistema de posicionamiento global)
- Contar con conexión a internet 4g

A.2 Instalación

Para iniciar con la instalación de la aplicación en nuestro dispositivo móvil procederemos a adquirir el siguiente archivo Apk (*Android aplication package*). en el departamento de porteo. Procederemos a transferir este archivo a nuestro dispositivo.

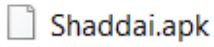


Figura A.1: Apk(*Android aplication package*) de la aplicación

Fuente: edición propia

Nos dirigimos a la carpeta donde alojamos este archivo en nuestro dispositivo móvil, a continuación, nos aparecerá de la siguiente manera como se muestra en la siguiente figura.

Figura A.2: Apk (*Android aplication package*) en dispositivo móvil

Fuente: edición propia

Procederemos a pulsar la aplicación, nos mostrar una ventana como se muestra en la siguiente figura, al ser una aplicación que no se encuentra alojada dentro de *play store* nos pedirá que le demos permiso al dispositivo de poder instalar esta Apk (*Android aplication package*).

Procedamos a seleccionar ajustes. A continuación, veremos los ajustes de pantalla de bloque y seguridad, nos dirigiremos a fuentes desconocidas.

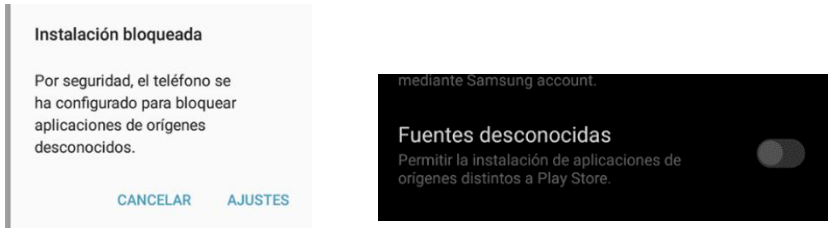


Figura A.3: configuración de fuentes desconocidas

Fuente: edición propia

A continuación, nos mostrara una ventana emergente donde nos da una advertencia de instalar aplicaciones de fuentes externas, dejaremos la configuración como se muestra en la siguiente figura y daremos en aceptar.

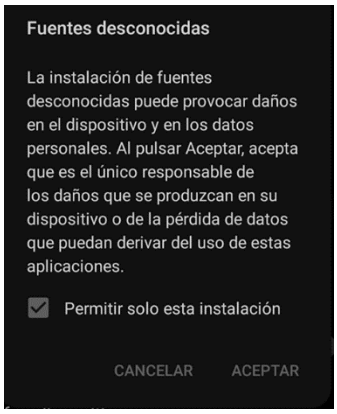


Figura A.4: ventana de permiso de fuentes desconocidas

Fuente: edición propia

A continuación, aparecerá la ventana de instalación de la aplicación como se muestra en la siguiente figura. Daremos en instalar para pasar al siguiente paso. Esperamos que se termine de instalar, Ya finalizada la instalación nos mostrar una pantalla como la siguiente, procederemos a seleccionar abrir.

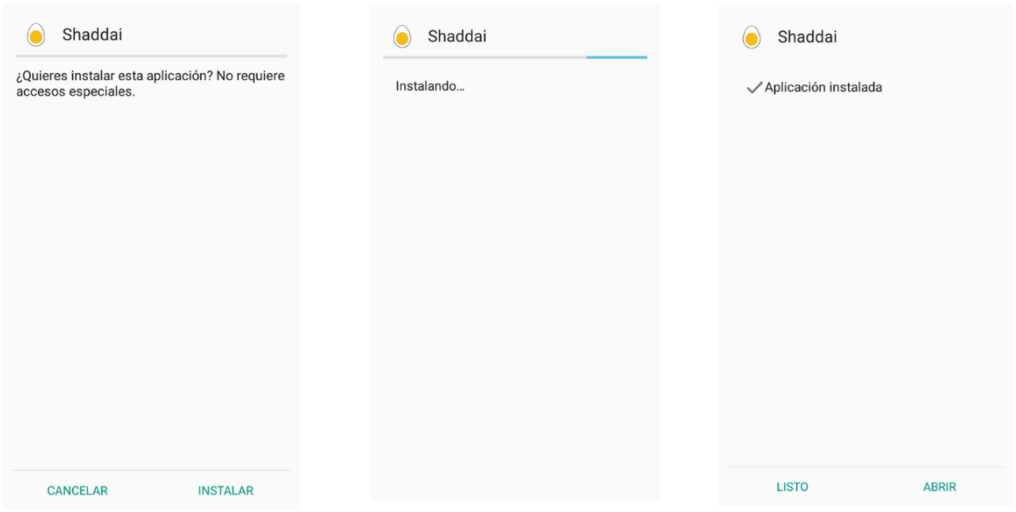


Figura A.5: finalización de la instalación

Fuente: edición propia

A continuación, nos mostrará la pantalla principal de la aplicación, saldrá una ventana emergente pidiéndonos que aceptemos los permisos que requiere la aplicación para su funcionamiento, aceptamos todos los permisos y finalmente ingresamos el nombre y apellido.

Figura A.6: ventana principal de la aplicación

Fuente: edición propia

Tendremos un acceso directo en nuestra pantalla de inicio como se muestra en la siguiente figura.

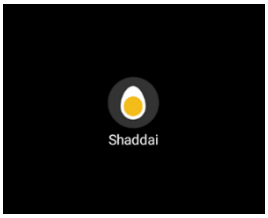


Figura A.7: acceso directo

Fuente: edición propia

A.3 Manual de usuario.

El presente manual sirve como apoyo de uso para la aplicación móvil de El Shaddai, con esta aplicación es posible que los operadores puedan

Registrarse.

Al ser la primera vez que se utiliza la aplicación se necesitara registrarse con sus credenciales, se procede a ingresar su nombre y su apellido paterno dentro de los campos de texto que se muestran en la siguiente imagen, al presionar el botón entrar, se desplegara una ventana emergente que funciona como asistente de registro de usuario. Ya finalizado este proceso la propia aplicación nos mostrara un mensaje avisándonos que nos hemos registrado con éxito.

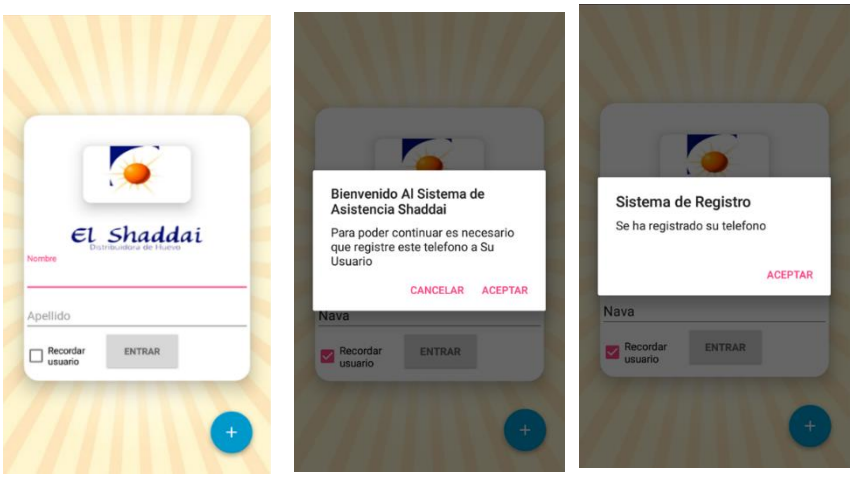


Figura A.8: registro

Fuente: edición propia

Sesión.

Al iniciar sesión por primera vez, se muestra la pantalla de inicio de sesión.

Para iniciar sesión en la aplicación, introduce tu nombre y apellido paterno escribiendo la primera letra en mayúscula en cada campo de texto, si los datos que han sido proporcionados son correctos pasará a la pantalla de menú donde podrá seleccionar las opciones disponibles.

En siguientes ocasiones que se ingrese a la aplicación, no será necesario que proporcione de nuevo sus credenciales nuevamente, bastara con solo presionar el botón de entrar.

Figura A.9: ventana principal

Fuente: edición propia

Registro de asistencia.

Ya iniciada la sesión dentro de la aplicación, para poder registrar nuestra hora de entrada será necesario seleccionar la opción asistencia como se muestra a continuación. A continuación, nos mostrará una pantalla como se ve en la siguiente imagen, dentro de esta podremos comenzar a registrar nuestras asistencias presionando el botón Registrar asistencia, posterior a esto se comenzará a reflejar la hora en que se generó la petición en el tablero que se encuentra en la parte inferior del botón de registro.

Figura A.10: asistencia

Fuente: edición propia

Remisiones

Ya iniciada la sesión dentro de la aplicación, para poder visualizar las remisiones que están a nuestro cargo será necesario seleccionar la opción Remisiones como se muestra a continuación en la figura A.11. Como se puede observar se da una breve información de las remisiones como es su cliente, sucursal, fecha de salida y el estatus de entrega en que se encuentra. En caso de necesitar consultar alguna remisión en específico se presiona en el icono de la lupa que se encuentra en la parte superior derecha de la pantalla y escribimos alguno de los campos que se muestran en nuestra lista para filtrarla.

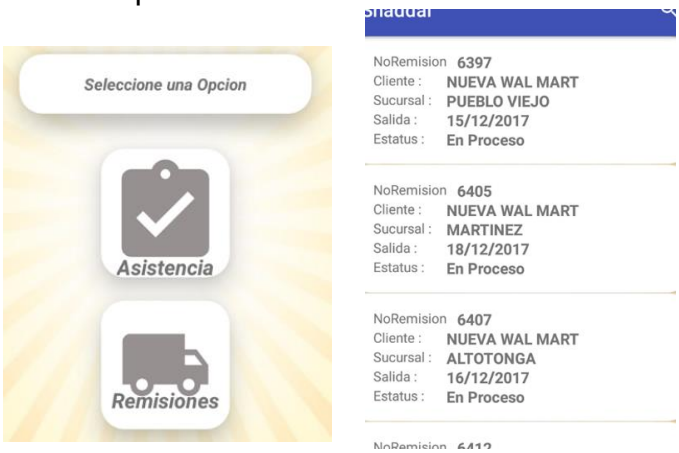


Figura A.11: lista de remisiones

Fuente: edición propia

Devolución sin piezas

Para capturar una devolución sin pieza seleccionamos alguno de los elementos que se muestran en la lista de remisiones, ahora nos desplegara una ventana emergente donde nos pide seleccionar entregado o devolución, seleccionamos entregado, posterior a esto se mostrara una ventana emergente que nos mostrara el número de folio de la remisión seleccionada y un cuadro de texto donde podremos ingresar el folio de entregado, se captura el folio y se procede a enviar y recibiremos un mensaje de respuesta.

Figura A.12: devolución sin pieza

Fuente: edición propia

Devolución con piezas.

Para capturar las devoluciones que contienen devolución en alguna de sus presentaciones procedemos a seleccionar la remisión en nuestra lista, una vez que se muestra la ventana emergente de opciones, seleccionamos devolución, a continuación, nos mostrara una ventana como se muestra a continuación, se procede a capturar el folio de devolución en la parte superior , para capturar las piezas devueltas podremos y la observación podremos ingresarlas en el tablero que se muestra en la parte inferior.

Figura A.13: devolución con pieza

Fuente: edición propia

Para finalizar con la devolución procedemos a presionar el botón capturar devolución, a continuación, nos mostrara una ventana de confirmación, daremos aceptar y finalmente tendremos un mensaje de que nuestra petición fue insertada con éxito.

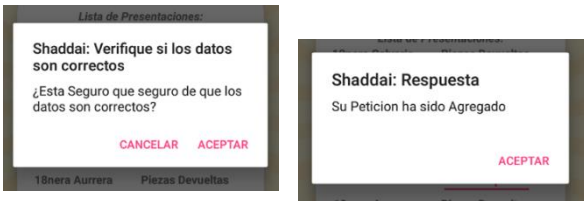


Figura A.14: respuesta de inserción

Fuente: edición propia