

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

TransPack-App móvil para la
gestión del transporte de
paquetes en Android

Curso 2017/2018

Alumno/a:

Mohamed el amine Mounaji

Director/es:

Rosa María Ayala Palenzuela



Agradecimientos

Esta memoria significa un punto y final a una de las mejores etapas de mi vida y me gustaría agradecer el apoyo recibido a todas aquellas personas que han estado a mi lado, en lo bueno y en lo malo, durante este tiempo.

Quiero agradecerle a mi directora de trabajo Rosa María Ayala Palenzuela, por su gran apoyo en este trabajo, por su ayuda y por sus consejos. A todos y cada uno de los profesores de la universidad de informática de Almería por todos los consejos y apoyo y por aprender de su gran experiencia.

Quiero agradecer de forma muy especial todo el apoyo que me han dado mis padres desde que comenzó la etapa universitaria, ya que sin ese apoyo que me han dado siempre, en todos los aspectos de la vida, no hubiera llegado hasta aquí.

Por último y especialmente a mi pareja Manale, por su apoyo y sus ánimos en los buenos y malos momentos. Gracias por estar siempre a mi lado.

Tabla de contenido

1. Resumen.....	1
2. Introducción.....	3
2.1 Justificación del Trabajo.....	5
2.2 Objetivos.....	6
2.3 Fases del desarrollo.....	7
2.4 Medios empleados.....	8
2.5 Planificación temporal.....	9
3. Estado del Arte.....	12
3.1 Introducción.....	12
3.2 Estudio de las plataformas móviles actuales.....	14
4. La plataforma Android.....	19
4.1 Introducción.....	19
4.2 Evolución de la plataforma.....	20
4.3 Arquitectura de Android.....	28
4.3.1 Nivel de aplicaciones.....	29
4.3.2 Nivel del framework de aplicaciones.....	29
4.3.3 Librerías.....	30
4.3.4 Tiempo de ejecución de Android.....	30
4.3.5 Kernel Linux.....	31
4.4 Aplicaciones.....	32
4.4.1 Componentes de una Aplicación.....	32
4.4.2 Ciclo de vida de una aplicación Android.....	33
4.5 Interfaz de Usuario Android.....	39
4.6 Recursos.....	41
4.7 Proveedores de contenido.....	41
4.8 El archivo AndroidManifest.xml.....	41
4.9 Permisos.....	42
5. Otras tecnologías usadas en el trabajo.....	44
5.1 Cliente-servidor.....	44
5.2 El protocolo HTTP.....	44
5.3 El lenguaje de programación Java al lado del cliente.....	45
5.4 MYSQL.....	45
5.5 Ubuntu server.....	45
5.6 LAMP.....	45
5.7 Webmin.....	46

5.8 PHPMyAdmin	47
6. Análisis, diseño e implementación de la solución	49
6.1 Análisis	49
6.2 Requisitos de usuario	49
6.2.1 Requisitos de capacidad.....	50
6.2.2 Requisitos de restricción.....	56
6.3 Casos de uso.....	58
6.3.1 Operaciones que pueden realizar todos los usuarios	58
6.4 Diagrama de clases.....	76
6.5 Diseño	78
6.5.1 Arquitectura del software	78
6.5.2 Diseño de la base de datos	80
7. Conclusiones y trabajos futuros.....	81
7.1 Conclusiones	81
7.2 Trabajos futuros	82
8. Presupuesto	84
8.1. Coste de material.....	84
8.2. Coste de mano de obra.....	84
8.3 Coste total.....	86
9. Resultados	87
10. Pruebas.....	101
10.1 Introducción	101
10.2 pruebas de la caja blanca o enfoque estructural	102
10.3 Pruebas de la caja negra o enfoque estructural	102
Bibliografía	107

Índice de tablas

Tabla 1: tabla de planificación del tfg.....	9
Tabla 2: COMPARATIVA DE PLATAFORMAS MOVILES.....	15
Tabla 3: Evolución de la plataforma Android [9]	27
Tabla 4: REQUISITOS DE CAPACIDAD (1).....	50
Tabla 5:REQUISITOS DE CAPACIDAD (2).....	50
Tabla 6: REQUISITOS DE CAPACIDAD (3).....	51
Tabla 7: REQUISITOS DE CAPACIDAD (4).....	51
Tabla 8: REQUISITOS DE CAPACIDAD (5).....	51
Tabla 9: REQUISITOS DE CAPACIDAD (6).....	51
Tabla 10: REQUISITOS DE CAPACIDAD (7).....	52
Tabla 11: REQUISITOS DE CAPACIDAD (8).....	52
Tabla 12: REQUISITOS DE CAPACIDAD (9).....	52
Tabla 13: REQUISITOS DE CAPACIDAD (10).....	52
Tabla 14: REQUISITOS DE CAPACIDAD (11).....	53
Tabla 15: REQUISITOS DE CAPACIDAD (12).....	53
Tabla 16: REQUISITOS DE CAPACIDAD (13).....	53
Tabla 17: REQUISITOS DE CAPACIDAD (14).....	54
Tabla 18: REQUISITOS DE CAPACIDAD (15).....	54
Tabla 19: REQUISITOS DE CAPACIDAD (16).....	54
Tabla 20: REQUISITOS DE CAPACIDAD (17).....	55
Tabla 21: REQUISITOS DE CAPACIDAD (18).....	55
Tabla 22: REQUISITOS DE CAPACIDAD (19).....	55
Tabla 23: REQUISITOS DE CAPACIDAD (1).....	56
Tabla 24: REQUISITOS DE CAPACIDAD (2).....	56
Tabla 25: REQUISITOS DE CAPACIDAD (3).....	56
Tabla 26: REQUISITOS DE CAPACIDAD (4).....	57
Tabla 27: CASO DE USO (1)	59
Tabla 28:CASO DE USO (2)	59
Tabla 29: CASO DE USO (3)	59
Tabla 30: CASO DE USO (4)	60
Tabla 31: CASO DE USO (5)	61
Tabla 32: CASO DE USO (6)	61
Tabla 33:CASO DE USO (7)	62
Tabla 34: CASO DE USO (8)	63
Tabla 35: CASO DE USO (9)	63
Tabla 36:CASO DE USO (10)	64
Tabla 37:CASO DE USO (11)	64
Tabla 38: CASO DE USO (12)	65
Tabla 39: CASO DE USO (13):	66
Tabla 40: CASO DE USO (14)	66
Tabla 41: CASO DE USO (15)	66
Tabla 42: CASO DE USO (16)	67
Tabla 43:CASO DE USO (17)	68
Tabla 44: CASO DE USO (18)	69
Tabla 45: CASO DE USO (19)	70
Tabla 46: CASO DE USO (20)	70
Tabla 47: Modificar transportista (21)	70

Tabla 48: Modificar transportista (22)	71
Tabla 49: Modificar transportista (23)	72
Tabla 50: Modificar transportista (24)	72
Tabla 51: Modificar transportista (25)	73
Tabla 52: Modificar transportista (26)	74
Tabla 53: Mostrar lista de paquetes de un viaje (27)	74
Tabla 54: Mostrar lista de paquetes de un viaje (28)	75
Tabla 55: Mostrar lista de recogidas del transportista (29)	75
Tabla 56: Tabla de costes de Material.....	84
Tabla 57: dias reales de trabajo	85
Tabla 58: Costes totales.....	86
Tabla 59: modelo de tabla de pruebas	102
Tabla 60: logueo correcto del usuario.....	102
Tabla 61: logueo incorrecto del usuario.....	103
Tabla 62: registro un cliente con datos validos	103
Tabla 63: registro de un cliente con datos no validos	103
Tabla 64: actualizar los datos de un cliente con datos no válidos	103
Tabla 65: actualizar los datos de un cliente con datos válidos.....	104
Tabla 66: búsqueda del cliente con datos correctos	104
Tabla 67: búsqueda del cliente con datos incorrectos	104
Tabla 68: registro de un usuario con datos validos.....	104
Tabla 69: registro de un usuario con datos invalidos	104
Tabla 70: Actualización de un usuario con datos no válidos.....	105

Índice de Figuras

Figura 1: digrama de gant	10
Figura 2: Algunos modelos de Smartphone.....	13
Figura 3: Tiendas de aplicaciones en iOS y Android	16
Figura 4: CUOTA DEL MERCADO DE SMARTPHONE EN ESPAÑA [7]	16
Figura 5: ARQUITECTURA DE ANDROID [10]	28
Figura 6: CICLO DE VIDA DE UNA ACTIVIDAD	35
Figura 7: BUCLE DE EJECUCIÓN DE SERVICIO	36
Figura 8: ÁRBOL JERÁRQUICO DE UNA UI EN ANDROID	39
Figura 9: ARCHIVO XML DE UNA ACTIVIDAD EN ANDROID	40
Figura 10: El archivo AndroidManifest.xml.....	42
Figura 11: PERMISOS DENTRO DEL MANIFEST	43
Figura 12:Figura descriptiva de la arquitectura cliente-servidor	44
Figura 13: LA INTERFAZ DE WEBMIN	46
Figura 14:phpmyadmin.....	47
Figura 15: Registrar usuarios	58
Figura 16: Iniciar sesión	60
Figura 17: Gestión de clientes	61
Figura 18: Gestión de viajes	62
Figura 19: Cerrar viaje.....	63
Figura 20: : Modificar transportista	64
Figura 21: Registrar un paquete	65
Figura 22:Añadir un paquete registrado al viaje	67
Figura 23: Pasar paquete a otro viaje.....	68
Figura 24: Entregar paquete.....	68
Figura 25: Crear nueva incidencia	69
Figura 26: Modificar transportista	70
Figura 27: Registrar futuras recogidas.....	71
Figura 28: Lista general de recogidas.....	71
Figura 29: Lista de viajes activos.....	72
Figura 30: Descubrir paquete	73
Figura 31: Descubrir paquete	73
Figura 32: Lista de mercancías de un viaje	74
Figura 33: Lista de entregas.....	74
Figura 34: Lista de recogidas del transportista	75
Figura 35: DIAGRAMA DE CLASES	77
Figura 36: Arquitectura del sofwatre	78
Figura 37: Diseño de la base de datos	80
Figura 38: Men u de aplicaciones del dispositivo.....	87
Figura 39: pantalla de bienvenida.....	87
Figura 40: Inicio de sesión.....	88
Figura 41: Primer menú principal.....	88
Figura 42: segundo menú principal	89
Figura 43:Registro de clientes	89
Figura 44: Actualización de los datos de un cliente	90
Figura 45: registro de usuario.....	90
Figura 46: Actualizar usuario.....	91
Figura 47: registro de recogida de paquetes	91

Figura 48:Lista general de recogidas.....	92
Figura 49: registro de paquetes.....	92
Figura 50: añadir paquete durante el viaje.....	93
Figura 51: pasar paquete a otro viaje	93
Figura 52: entrega de paquetes.....	94
Figura 53: Añadir incidencia	94
Figura 54: modificar transportista	95
Figura 55: Crear viaje	95
Figura 56:cerrar viaje	96
Figura 57: añadir coste.....	96
Figura 58: viajes activos	97
Figura 59: descubrir paquete.....	97
Figura 60: lista entregas	98
Figura 61: lista incidencias	98
Figura 62: lista de recogidas del transportista	99

1. Resumen

Hoy en día, en la era de Internet y de las comunicaciones, las empresas necesitan que sus procesos sean ágiles y eficientes, y tan automatizados como sea posible, ya que no existe tiempo que perder. Teniendo en cuenta lo anterior, este Trabajo de Fin de Grado se centra en optimizar uno de los procesos más importantes de cualquier empresa de transporte, la gestión y trazabilidad de transporte de mercancías y paquetes.

Otro factor que ha motivado la realización de este trabajo es la aceptación y expansión que tienen hoy en día los dispositivos móviles inteligentes (smartphones), y más aún, en el ámbito empresarial. La gran capacidad de procesamiento, almacenamiento y personalización con la que cuentan estos dispositivos, hace posible que hoy en día cualquier trabajador de cualquier corporación pueda llevar casi la totalidad de la información necesaria para la realización de su trabajo a donde quiera que vaya, sin tener que estar físicamente en su puesto de trabajo. Esto deriva a una gran flexibilidad y agilidad en los procesos laborales, ahorrando costes y gran cantidad de tiempo. Así pues, se desean aprovechar todas estas características para su aplicación en el proceso de distribución y recogida de mercancías.

Finalmente, otro aspecto fundamental para que este trabajo se lleve a cabo de manera viable, es el gran avance en infraestructuras de red y descentralización de la información fundamental con la que cuentan las empresas hoy en día. Esto hace posible acceder de manera fiable y segura a la información sobre cada mercancía en reparto y recogida ahorrando tiempo y recursos.

En resumen, todos los factores expuestos hacen perfectamente viable, e incluso necesaria, la realización de este Trabajo existe la necesidad, se cuenta con los recursos necesarios para su desarrollo y no se detectan factores importantes que limiten su aplicación.

2. Introducción

La tecnología celular ha experimentado un gran avance desde el lanzamiento al mercado del primer teléfono móvil en 1983 hasta los terminales de los que disponemos hoy en día. Las distintas necesidades, así como los avances logrados dieron lugar a generaciones tecnológicas bien diferenciadas cuyas capacidades marcaron a su vez la evolución tanto del hardware como del software de los terminales. Dichas generaciones resumidas brevemente son las siguientes [1][2]:

Generación 1G

- La transferencia analógica y estrictamente para voz son características identificadoras de esta generación. La calidad de los enlaces de voz era muy baja y la velocidad de conexión no era mayor a 2400 bauds. Basadas en FDMA, había una limitación en el número de usuarios que podrían usar el servicio simultáneamente. Además, la seguridad era inexistente.

Generación 2G

- La mejora más importante que aportó esta tecnología es que las transferencias son digitales. Uso a su vez TDMA para permitir que hasta ocho usuarios utilizaran los canales. Los protocolos empleados en los sistemas 2G soportan velocidades de información más altas para voz, pero limitados en comunicaciones de datos. Se pueden ofrecer otros servicios tales como datos, fax y SMS. Las tecnologías predominantes son: GSM, IS-136, CDMA y PDC.

Generación 2.5G

- La entrada del GPRS en escena se considera la evolución del 2G al 3G. El GPRS puede usarse para servicios WAP, SMS, MMS y para servicios de comunicación por Internet como el email y el acceso a la web. Con este nuevo sistema los canales pasaron a compartirse por más de un usuario lo que permita comunicaciones más eficientes y baratas.

Generación 3G

- Se conoce a esta generación como un conjunto de estándares de telecomunicaciones. Los servicios 3G permiten el uso de voz y datos simultáneamente a altas velocidades, lo que hace esta generación apta para aplicaciones multimedia y altas transmisiones de datos. La tecnología usada en estas redes es UMTS.

Una vez conocida la evolución de la tecnología celular y las posibilidades que ofrecía en cada momento, podemos entender mejor la evolución sufrida por los terminales móviles que fueron adaptándose de forma que aprovecharan dichas posibilidades.

Generación 4G

El sistema móvil de cuarta generación está basado totalmente en IP. El objetivo principal de la tecnología 4G es proporcionar alta velocidad, alta calidad, alta capacidad, seguridad y servicios de bajo coste para servicios de voz y datos, multimedia e internet a través de IP. Para usar la red de comunicación móvil 4G, los terminales de los usuarios deben ser capaces de seleccionar el sistema inalámbrico de destino. Para proporcionar servicios inalámbricos en cualquier momento y en cualquier lugar, la movilidad del terminal es un factor clave en 4G.

Generación 5G

La capa física y de enlace de datos define la tecnología inalámbrica 5G indicando que es una tecnología Open Wireless Architecture (OWA).

Para realizar esto, la capa de red está subdividida en dos capas, capa de red superior para el terminal móvil y un menor nivel de red para la interfaz. Aquí todo el enrutamiento se basa en direcciones IP que serían diferentes en cada red IP en todo el mundo.

En la tecnología 5G la pérdida de velocidad de bits se supera mediante el Protocolo de Transporte Abierta (OTP). El OTP es soportado por transporte y capa de sesión. La capa de aplicación es para la calidad de la gestión de servicio a través de varios tipos de redes.

2.1 Justificación del Trabajo

Dado el creciente número de empresas de tamaño pequeño y mediano dedicados al transporte de paquetes entre Europa y Marruecos y entre estas cabe destacar una empresa familiar que se dedica a la misma actividad entre Alemania y Marruecos, surge la idea de desarrollar una aplicación móvil a medida que gestiona todo el proceso y las necesidades de transporte de paquetes.

Las principales ventajas que se pueden obtener mediante la implantación de un sistema de estas características en dichas empresas son: la reducción de costes, la mejora de la calidad del servicio, aumentar la rentabilidad y beneficios.

Para negocios en los que la rapidez es clave, poder reaccionar ante imprevistos de cualquier tipo de manera inmediata es vital para aumentar la satisfacción y la confianza de la clientela, cuya exigencia es cada vez mayor y demandan una gestión óptima en los envíos. Con este tipo de soluciones se tiene la mayor garantía de poder conseguirlo.

En resumen, esta aplicación aunaría en una sola herramienta todas las funcionalidades necesarias. Además, el hecho de desarrollarse sobre la plataforma Android haría a la aplicación muy escalable, no limitando la funcionalidad a la comunicación entre el dispositivo y el servidor, sino que el usuario dispondría de una herramienta muy potente gracias a la posibilidad de integrar fácilmente otros módulos como por ejemplo información sobre meteorología, talleres, tráfico, etc. Este es un punto que hay que tener en cuenta de cara al futuro, ya que Android es una plataforma que está creciendo a gran velocidad gracias al aporte tanto de aplicaciones de terceros como por parte del propio Google.

2.2 Objetivos

El objetivo de este TFG es lograr el diseño y desarrollo de una aplicación compuesta por varios módulos intercomunicados entre sí que permitan agilizar y mejorar la actividad de gestión y trazabilidad en una empresa de transporte de mercancías.

Para alcanzar este objetivo principal, existen otros objetivos secundarios a conseguir:

- Análisis previo de los procesos principales en la gestión de transporte de mercancías, que serán implementados en el sistema.
- Análisis de la información necesaria que se almacenará en una base de datos, que actuará como base a una base de datos corporativa real.
- Diseño e implementación de la base de datos que contemple la información obtenida en el punto anterior [25] [26].
- Diseñar e implementar una aplicación en Android para la gestión de transporte de mercancías, que se comunique con el servidor (Cliente-Servidor) de manera eficaz y eficiente mediante la tecnología web service.
- La aplicación permite generar y leer códigos QR para cada mercancía, de esta manera ahorramos tiempo, facilitamos y organizamos el trabajo.
- Se implementará un módulo de servicios web con el objetivo de enviar a los terminales móviles los datos procedentes de la base de datos, así como recoger y almacenar en la base de datos la información recibida por dichos terminales.
- Conocer las principales características de Android [23] [24]. El primer paso para conocer este sistema debe consistir en indagar toda la información posible sobre él, a fin de conocer cuál es su arquitectura, sus componentes básicos, y cuál es su comportamiento al ejecutar las aplicaciones, consultando la documentación pertinente.

2.3 Fases del desarrollo

Consideraremos las siguientes fases del desarrollo para nuestro trabajo, basadas en el modelo en cascada, es decir, de forma secuencial:

Análisis de requisitos

Extraer los requisitos debe ser el primer paso. El desarrollador debe comprender la naturaleza del problema, las necesidades del cliente, la función requerida, así como el comportamiento de la aplicación a realizar. Para ello se debe desarrollar un catálogo de requisitos que recoja dichos aspectos.

Diseño

Determinar el comportamiento esperado de la aplicación, es decir, el funcionamiento básico sin entrar en detalles.

Implementación

Reducir el diseño a código. La complejidad y la duración la determinará en mayor o menos medida el lenguaje de programación utilizado.

Pruebas

Comprobar que el software realice correctamente las tareas indicadas en el diseño y del catálogo de requisitos.

Documentación

Modelaciones (UML) [27], diagramas, implementación, pruebas, resultados ...

2.4 Medios empleados

- **Android Studio** que es el entorno de desarrollo integrado oficial para la plataforma Android, Fue anunciado el 18 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android.
- El Portátil **Lenovo G50** que lleva la tecnología i7, 8Gb de memoria RAM, Disco duro de 1 TB
- **Nexus 7 Vernee Mix 2** para hacer pruebas.
- **Ordenador Hp** sobre mesa usado como servidor para poder alojar todos los servicios de la aplicación.
- Servicio de alojamiento de archivos multiplataforma **OnoDrive**.
- **Microsoft office 365 ProPlus**.
- **Notpad++** para crear código en php
- **Phpmyadmin**
- El control de versiones **Git** [\[22\]](#)
- **Webmin** parala gestión del servidor
- **Ubuntu server**
- Windows 10 education

Dado que el trabajo consiste en una aplicación de tipo cliente-servidor, la parte servidora del mismo se ha suplido con el mismo equipo de desarrollo. Así mismo, para la depuración y pruebas de la aplicación cliente se ha utilizado un teléfono móvil Vernee Mix 2 Android 7.0 y un Tablet Nexus 7 con Android 6.0, además del emulador con el que cuenta Android studio.

2.5 Planificación temporal

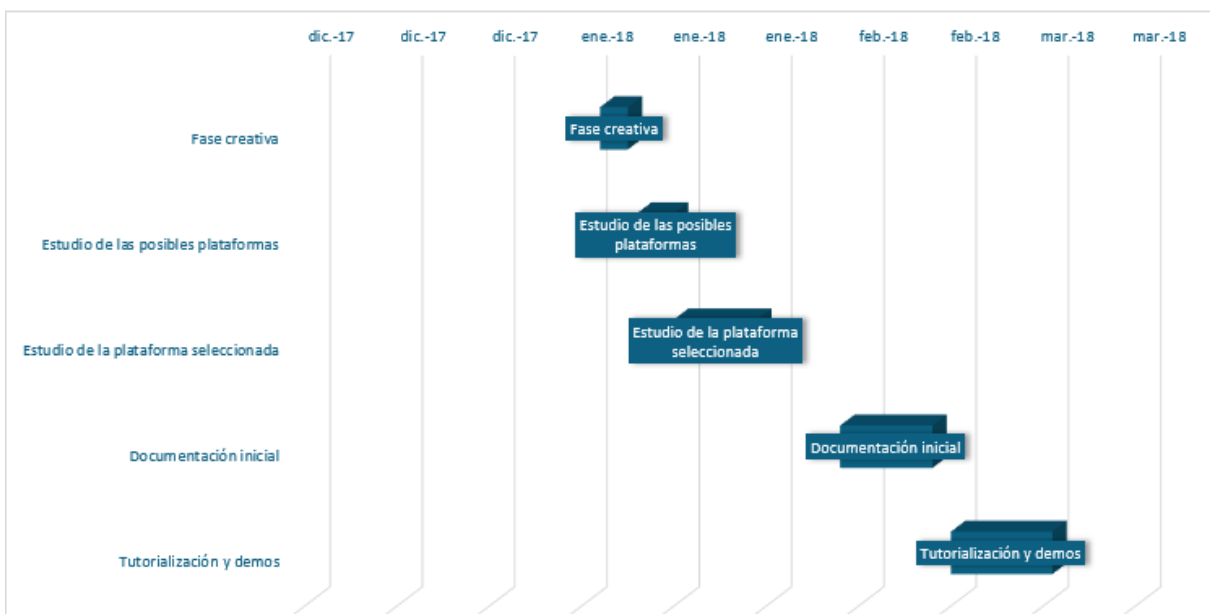
Antes de comenzar el TFG, es necesario realizar una planificación que permita dividirlo en distintas fases. Gracias a esta planificación en etapas del proyecto, se consigue hacer un seguimiento del mismo y así comprobar si se están cumpliendo las fechas comprometidas en un primer momento.

A continuación, se va a detallar la planificación seguida para la elaboración del trabajo. En la siguiente tabla se pueden ver las actividades realizadas durante la elaboración, indicándose la fecha de inicio y finalización

Actividad	Fecha inicio	Fecha fin	Duración (días)
Fase creativa	12/01/2018	14/01/2018	2
Estudio de las posibles plataformas	16/01/2018	19/01/2018	3
Estudio de la plataforma seleccionada	20/01/2018	28/01/2018	8
Documentación inicial	07/02/2018	16/02/2018	9
Tutorialización y demos	19/02/2018	01/03/2018	10
Diseño de la aplicación	05/03/2018	13/03/2018	8
Implementación	15/03/2018	01/06/2018	65
Documentación	02/06/2018	07/08/2018	65
Pruebas	08/08/2018	20/08/2018	12
Total	12/01/2018	20/08/2018	182

TABLA 1: TABLA DE PLANIFICACIÓN DEL TFG

La siguiente figura muestra el diagrama de Gantt asociados a la planificación descrita en la tabla anterior:



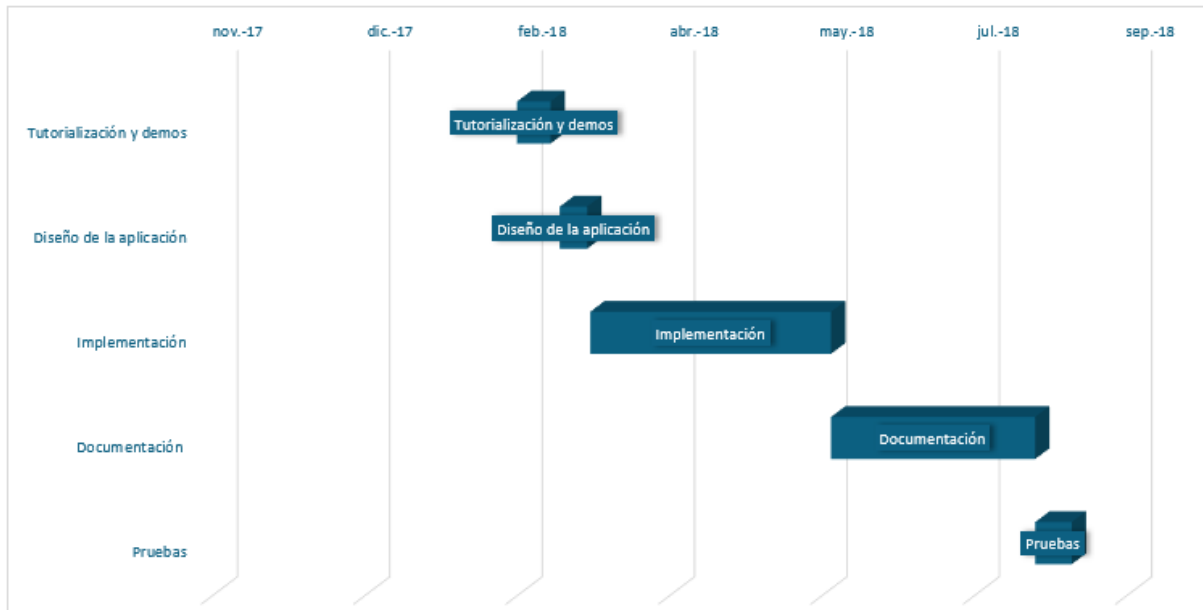


FIGURA 1: DIGRAMA DE GANT

3. Estado del Arte

3.1 Introducción

Hoy en día hablar del término dispositivo móvil puede dar lugar a una gran variedad de interpretaciones distintas, ya que existen multitud de aparatos electrónicos que poseen muchas de las características que se le presuponen a un dispositivo de este tipo.

Si tenemos en cuenta una definición más o menos rigurosa de este término como es la siguiente: “aparatos de pequeño tamaño, con algunas capacidades de procesamiento altas, móviles o no, con conexión permanente o intermitente a una red, con memoria aceptable, diseñados específicamente para una función, pero que pueden llevar a cabo otras funciones más generales”, podemos comprobar que como se mencionó anteriormente hay una gran variedad de aparatos que cumplen esta definición, en mayor o menor medida, como por ejemplo una PDA, un teléfono móvil, un lector de libros electrónico o un ordenador portátil, una Tablet.

De entre los elementos mencionados destaca, sin lugar a dudas, el teléfono móvil como el dispositivo más utilizado de entre todos. No obstante, en España la media de teléfonos móviles por persona es superior a un teléfono por habitante. Si vamos más allá de las fronteras de España, y nos fijamos en los datos a nivel mundial, El número de usuarios únicos de telefonía móvil alcanzó los 5.000 millones al finalizar 2017, lo que supone un grado de penetración del 66%, aunque el número de tarjetas SIM usadas por personas (excluyendo las que usan máquinas entre sí) se elevó a 7.800 millones, el 103% de los habitantes del planeta, superando así por primera vez la población mundial (7.600 millones de personas). Son datos del informe anual *Mobile Economy* de la GSMA, la asociación que organiza el Mobile World Congress (MWC) que se celebra en Barcelona [3].

Teniendo presentes estos datos es lógico pensar que el mercado de la telefonía móvil resulta muy interesante desde un punto de vista económico o de oportunidad de negocio. Como ingeniero informático, una de estas oportunidades a las que hago referencia es desarrollar software para este tipo de dispositivos.

Es evidente que no todos los teléfonos móviles tienen las mismas características o permiten realizar las mismas funciones. Si bien hace unos años los teléfonos servían exclusivamente para hablar y mandar SMS, hoy en día un móvil te permite entre otras cosas reproducir contenidos multimedia, conectarte a internet desde cualquier punto, sacar fotografías, utilizarlo como dispositivo GPS, etc. La presencia de esta serie de características añadidas en un teléfono móvil ha hecho que se empiece a utilizar un término nuevo para referirnos a estos aparatos.




Nace así el concepto de “smartphone” o en castellano “teléfono inteligente” para denominar a los teléfonos móviles que ofrecen estas capacidades (acceso al correo, internet, etc.) y que además suelen presentar otros atributos físicos como son la presencia de pantallas táctiles que permiten manejar el teléfono o la incorporación de teclados QWERTY [4] como dispositivos de entrada de texto.



FIGURA 2: ALGUNOS MODELOS DE SMARTPHONE

Todas estas capacidades o funciones mencionadas anteriormente estarían desaprovechadas sin software que las saque partido. Es por ello que en la actualidad oímos hablar de las tiendas o servicios de venta en línea de aplicaciones para teléfonos móviles como el “App Store” de Apple [5] o el “Play Store” de Google [6]. Estos servicios se caracterizan por ofrecer al usuario un inmenso catálogo de aplicaciones, algunas gratuitas y otras de pago.

3.2 Estudio de las plataformas móviles actuales

CARACTERISTICAS	 Android	 IOS	 WINDOWS PHONE
Compañía	Google	Apple	Windows
Núcleo del SO	Linux	Mac OS X	Windows CE
Lenguaje de programación	Java, C++, Kotlin	Objective-C, C++	C#, muchos
Licencia de software	Software libre y abierto	propietaria	propietaria
Año de lanzamiento	2008	2007	2010
Motor del navegador	WebKit	WebKit	Pocket internet Explorer
Soporte Flash	Si	No	No
HTML5	Si	Si	Si
Tienda de aplicaciones	Play Store	App Store	Microsoft Store
Numero de aplicaciones	3.8000.000	3.500.000	250.000

Coste para publicar	\$25 una vez	\$99 / año	\$99/año
Plataforma de desarrollo	Windows, Mac, Linux	Mac	Windows
Actualizaciones automáticas de SO	Depende del fabricante	Si	Depende del fabricante
Fabricante Único	No	Si	No
Aplicaciones Nativas	Si	Si	No

TABLA 2: COMPARATIVA DE PLATAFORMAS MOVILES

Analizando los datos de la anterior tabla, la primera impresión es que sobre el papel las cuatro plataformas no difieren mucho en sus principales características, si bien hay ciertos detalles que las hacen más atractivas a unas sobre otras desde el punto de vista del usuario y también desde el punto de la vista de desarrollador.

Por un lado, desde la perspectiva del usuario habitual de la telefonía móvil, estos se suelen centrar en aspectos como la capacidad de personalización de la interfaz de usuario o lo bien que ve esta, si tienen un montón de aplicaciones gratuitas, o si la plataforma cuenta con un amplio catálogo de juegos.

Pese a la tan cacareada segmentación del sistema móvil de Google (cada fabricante de smartphones incluye su personalización), Android tiene una apariencia "muy limpia" visualmente hablando, y cuenta con las mejores apps de mapas, correo, mensajería, redes sociales o buscadores, instaladas de serie. Aquí es donde se nota la mano de Google, con sus maps, gmail, google+ y demás software sobresaliente, uno de los verdaderos pilares del éxito de este sistema.

El caso es que cualquier usuario que se inicie con los móviles inteligentes, necesitará más horas de aprendizaje para dominar un Android que si lo situamos frente a un terminal iOS o Windows Phone. Este punto no debe interpretarse negativamente, porque ese mismo usuario quedará encantado con las enormes posibilidades de personalización que le ofrece el sistema de Google, algo que en sistemas más cerrados como iOS es inimaginable por el momento.

Windows Phone 10 quizá sea el sistema más sencillo de utilizar para un usuario que no haya tocado un smartphone en su vida. La disposición de los iconos de las apps y las ventanas activas ("Live tiles") dejan atrás el clásico concepto de carpetas y archivos de los primeros

Windows para equipos de sobremesa. La personalización es otro aspecto que se ha cuidado al detalle en WP 10: podremos cambiar el menú a nuestro gusto con dos toques, para destacar las apps que más vayamos a utilizar, sin complicaciones.

Play Store y App Store cuentan con una gran cantidad de aplicaciones, gratuitas y de pago, y también con un extenso catálogo de juegos, de hecho, en la actualidad la App Store cuenta con más de 2.2 millones aplicaciones y juegos, y por su parte el Play Store de Android cuenta con cerca de 3 millones. No obstante, hay que mencionar que en ambas plataformas hay numerosas aplicaciones que no presentan un mínimo de calidad o tienen como objetivo el simple hecho de gastar una broma.



FIGURA 3: TIENDAS DE APLICACIONES EN IOS Y ANDROID

Cambiando el punto de vista al de un desarrollador de software, y relacionado con lo anterior, resulta evidente que en ambas plataformas existe una gran oportunidad de negocio, pues cada una de ellas cuenta con millones de usuarios. Por tanto, la siguiente pregunta a resolver es “¿Cuál de las dos?”. Para ello se van a discutir varios aspectos como la cuota de mercado, las facilidades de desarrollo, etc.

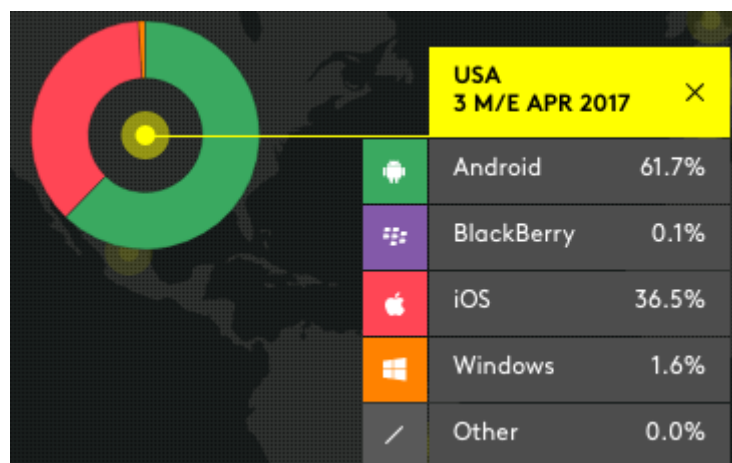


FIGURA 4: CUOTA DEL MERCADO DE SMARTPHONE EN ESPAÑA [7]

El motivo principal de la caída de iOS es su excesivo precio en sus dispositivos. Tener un iPhone con casi el mismo tamaño de aplicaciones que Android (vs) comprarte un Android de bajo coste con rendimiento bueno es lo que ha provocado que el sistema operativo de los de Google tenga casi el monopolio español lo mismo se puede decir para Marruecos.

Con estos datos se refuerza aún más la idea inicial de realizar el trabajo en la plataforma, Android pues está pensada exclusivamente para el mercado de smartphones y es la que tiene más cuota del mercado en todo el mundo.

A parte de todo lo mencionado anteriormente se deben tener en cuenta otras consideraciones para determinar la plataforma elegida y que están relacionadas con el propio proceso de desarrollo de aplicaciones para las mismas.

Todas estas plataformas, cuentan con bastantes herramientas de desarrollo, extensas APIs, multitud de librerías que extienden las funcionalidades de la plataforma y numeroso código fuente de ejemplo.

Sin embargo, desde mi punto de vista, Android cuenta con ciertas ventajas competitivas como poder desarrollar de forma independiente a la plataforma ya que en ningún caso requiere de hardware específico para el desarrollo, por el contrario, iOS (por ejemplo) requiere contar con un equipo que tenga instalado Mac OS X; o el coste de la licencia de desarrollador siendo más baja para Android.

Además, Apple ha establecido una política de privacidad bastante restrictiva, en la que hasta hace poco no se permitía a los desarrolladores hablar sobre el SDK o compartir código, sin mencionar que se deben obtener varios certificados emitidos por la compañía de la manzana incluso para probar las aplicaciones desarrolladas en nuestro iPhone. Por el contrario, Google con Android adopta una política más abierta facilitando mucho más el desarrollo de aplicaciones en lo relativo a estos aspectos.

Por último, otras consideraciones relativas al desarrollo, como el lenguaje de programación para cada plataforma dependerán de la experiencia de cada persona, y en mi caso particular, prefiero programar en java [28] por tener varias ventajas a los otros lenguajes de programación.

En conclusión, y en base a todo lo expuesto, se ha escogido Android para llevar a cabo este trabajo por las siguientes razones:

- Oportunidad de negocio
- Política de desarrollo menos restrictiva
- Inferiores costes de desarrollo (equipamiento y licencia)
- Lenguaje de programación y herramientas

4. La plataforma Android

4.1 Introducción

Android es una plataforma de software, pensada especialmente para dispositivos móviles, y que en pocas palabras se compone de un sistema operativo, un conjunto de aplicaciones base y una capa middleware situada entre las aplicaciones y el propio sistema operativo.

Dichas aplicaciones base están desarrolladas utilizando el lenguaje de programación Java, mientras que por otro lado ciertos componentes de la arquitectura de Android como las librerías o el núcleo, basado en el kernel de Linux, están escritos en C/C++.

En los siguientes apartados serán descritos distintos aspectos de la plataforma como su arquitectura, los componentes de una aplicación, etc., haciendo hincapié en los aspectos a priori más importantes. Toda esta información ha sido obtenida de la página oficial de desarrollo de Android, que ha sido traducida y resumida para incorporarla a este trabajo.

No obstante, y de forma introductoria, serán expuestas las principales características de Android

- Cuenta con un framework de aplicaciones que permite reutilizar o sustituir las aplicaciones existentes.
- Cuenta con una máquina virtual especialmente optimizada para dispositivos móviles conocida como Dalvik VM.
- Navegador integrado basado en el motor de código abierto WebKit [7].
- Capaz de procesar gráficos en 2D (gracias a la librería SGL) y gráficos 3D basados en la especificación OpenGL.
- Soporte para el almacenamiento de datos estructurados mediante las librerías SQLite.
- Soporte de múltiples formatos multimedia tanto de audio como video o imagen (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- Telefonía GSM.
- Comunicaciones siguiendo los protocolos estándar Bluetooth, EDGE, 3G, 4G y Wifi.
- Soporte de múltiples dispositivos hardware como cámaras, GPS, brújula o acelerómetros.
- Potente entorno de desarrollo que incluye un emulador, control de versiones Git

4.2 Evolución de la plataforma

Versiones	Novedades y Características
Apple Pie (v1.0)	<p>Android 1.0 Apple Pie fue la primera versión de Android comercial lanzada y este lanzamiento se realizó junto al HTC Dream el 23 septiembre de 2008.</p> <p>Las novedades e incorporaciones más destacadas de Android 1.0 fueron:</p> <ul style="list-style-type: none">• Incorporación de un mercado para compra y descarga de aplicaciones bajo el nombre de Android Market.• Un navegador web con soporte múltiples ventanas y capaz de abrir páginas web en HTML y XHTML.• Soporte básico para cámara de fotos.• Posibilidad de crear carpetas e introducir iconos de aplicaciones en ellas desde el escritorio• Acceder a servidores de correo electrónico por web soportando los protocolos POP3, IMAP4 y SMTP.• Sincronización con los productos de Google: Gmail, Google Calendar y Google Contacts.• Incorporación de productos de Google: GTalk, Google Maps, YouTube, Google Sync y Google Search.• Mensajería instantánea, SMS y MMS.• Reproductor de música (sin soporte para reproducir vídeo).• Soporte para teléfonos con LED.• Notificaciones en la barra de estado con posibilidad de establecer alertas por timbre, LED o vibración.• Marcación por voz.• Soporte para fondos de pantalla y Widgets.• Conectividad para WiFi y Bluetooth.

<p>Cupcake (v1.5)</p>	<p>Debido a las grandes mejoras introducidas en la tercera versión de Android el número de versión saltó directamente a la 1.5 desde la 1.1. Basado en el <i>kernel</i> Linux 2.6.27, las novedades más interesantes eran las siguientes:</p> <ul style="list-style-type: none"> • Rediseño completo de todos los elementos de la interfaz. • Transiciones animadas entre ventanas. • Mejoras en la velocidad de la cámara. • Menor tiempo de búsqueda de los satélites GPS, gracias a la posibilidad de utilizar A-GPS. • Mejoras en la velocidad del navegador web gracias a la inclusión de la última versión del motor de renderizado WebKit así como del intérprete de JavaScript SquirrelFish. • Añadida la posibilidad de copiar y pegar texto, así como de poder buscar texto dentro de una página web. • Posibilidad de personalizar los elementos mostrados en el escritorio. • Inclusión de teclado en pantalla, con soporte para orientación vertical y apaisada, funcionalidades de auto corrección y soporte de diccionarios del usuario. • Añadida la posibilidad de grabar y reproducir vídeos. • Añadido soporte Bluetooth A2DP y AVRCP.
<p>Donut (v1.6)</p>	<p>Lanzada en Septiembre de 2009, está basada en el <i>kernel</i> de Linux 2.6.29. Se considera una actualización menor, pero aun así se introducen algunas novedades interesantes:</p> <ul style="list-style-type: none"> • <i>Quick Search Box</i>, motor de búsqueda en la pantalla de inicio que permite buscar entre distintas fuentes (contactos, historial del navegador, Google). Con autocompletado y capacidad de aprendizaje. • Mejorada la velocidad de la cámara. • Posibilidad de conectarse a redes VPN, 802.1x. • Nuevas opciones de configuración para controlar el uso de la • batería, que permite comprobar qué aplicaciones y servicios son los que más consumen. Desde esta pantalla se puede también parar o desinstalar las aplicaciones instaladas en el dispositivo.

	<ul style="list-style-type: none"> • Las aplicaciones de <i>Android Market</i> aparecen ahora ordenadas por categorías (Aplicaciones, Juegos y Descargas). Para cada categoría podemos consultar las últimas actualizaciones y las aplicaciones más populares. Además, para cada aplicación se muestran capturas de pantalla y opiniones de otros usuarios. • Nuevo motor de texto a voz. • Añadidas herramientas para la creación de gestos (patrones dibujados en pantalla para realizar tareas específicas como añadir un marcador al navegador o copiar y pegar texto).
Éclair (v2.0/v2.1)	<p>En noviembre de 2009 se lanza la versión 2.0, continuando con la tradición de utilizar dulces de repostería como nombres para las versiones, las novedades más importantes de <i>Eclair</i> son:</p> <ul style="list-style-type: none"> • Optimizaciones para mejorar el rendimiento de los dispositivos. • Mejoras en la interfaz de usuario. • Rediseño de la interfaz del navegador, contando ahora con soporte para distintas características de HTML5 (entre ellas la etiqueta vídeo), la posibilidad de hacer zoom con una doble pulsación y miniaturas de los marcadores. • Soporte nativo de flash para la cámara. • Añadido zoom digital, modo escena, balance de blanco, efectos de color y modo macro. • Mejoras en el teclado virtual. • Soporte para nuevos tamaños y resoluciones de pantalla. • Rediseño de los contactos. • Bluetooth 2.1. • Soporte nativo de aplicaciones como Facebook. • Mejoras en Google Maps, que pasaba a ser multi-táctil y soportar capas de visionado. • Soporte de Microsoft Exchange. • Mejoras en el calendario. <p>En diciembre de 2009 se publicó una pequeña revisión, Android 2.0.1, que mejoraba la duración de la batería y la estabilidad, añadía la funcionalidad de llamada a tres, mejoras en el GPS, el Bluetooth, y la velocidad de disparo y auto foco de la cámara.</p>

	<p>Android 2.1, es lanzado en enero de 2010, también se considera una actualización menor, aunque incluye mejoras importantes:</p> <ul style="list-style-type: none"> • Reconocimiento de voz pudiéndose dictar frases o palabras en lugar de escribirlas en cualquier campo de texto. • Mejoras en el teclado virtual. • Galería de imágenes 3D. • Añadidos nuevos gestos para hacer zoom en el navegador, la galería y en Google Maps. • Rediseño de aplicaciones e inclusión de algunas nuevas como la aplicación de tiempo y noticias o de realidad aumentada como Google Googles. • Mejoras en Google Maps como la sincronización de nuestros sitios favoritos, el modo noche y auto completado de búsquedas. • Mejoras en la duración de la batería.
<p>Froyo (v2.2)</p>	<p>Lanzado en Junio de 2010, <i>Froyo</i> cuentas con las siguientes características:</p> <ul style="list-style-type: none"> • Actualizaciones automáticas para las aplicaciones. • Soporte Wifi 802.11n. • Soporte para Radio FM. • Soporte Flash 10.1 y Adobe AIR 2.5. • Soporte OpenGL 2.0. • Inclusión de un compilador JIT que mejora entre 2 y 5 veces el rendimiento de la versión 2.1. • Posibilidad de utilizar el dispositivo como modem utilizando la conexión USB así como de punto de acceso Wifi. • Incorporación del motor de JavaScript V8 del navegador Chrome. • Posibilidad de mover una aplicación instalada desde el teléfono a la tarjeta de memoria, y viceversa. • Opciones avanzadas de gestión energética. • Incluidas nuevas opciones de pago de aplicaciones como PayPal y Google Check Out.

<p>Gingerbread (v2.3)</p>	<p>Android 2.3 Gingerbread, es la actualización del sistema operativo Android 2.2 Froyo, fue lanzado el 6 de diciembre del 2010 y está basado en el Kernel de Linux 2.6.35.7 Actual, entre sus nuevas características podemos destacar las siguientes funciones:</p> <ul style="list-style-type: none"> • Soporte para dispositivos móviles • Actualización del diseño de la interfaz de usuario • Soporte para pantallas extragrandes y resoluciones WXGA y mayores • Soporte nativo para telefonía VoIP SIP • Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC • Nuevos efectos de audio como reverberación, ecualización, Soporte para Near Field Communication • Funcionalidades de cortar, copiar y pegar disponibles a lo largo del sistema • Teclado multi-táctil rediseñado • Soporte mejorado para desarrollo de código nativo • Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos • Recolección de elementos concurrentes para un mayor rendimiento • Soporte nativo para más sensores (como giroscopios y barómetros) • Un administrador de descargas para descargar archivos grandes • Administración de la energía mejorada y control de aplicaciones mediante el administrador de tareas • Soporte nativo para múltiples cámaras • Cambio de sistema de archivos de YAFFS a ext4
<p>Honeycomb (v3.0/v3.1/v3.2)</p>	<p>Con un lanzamiento previsto para Octubre de 2010, de esta versión se conocen aún muy pocos detalles, aunque si se ha confirmado que requerirá de dispositivos de altas prestaciones (CPU a 1 GHz, 512 MB de RAM, pantallas de 3,5 pulgadas o superiores). Las mejoras que se espera que incluya son las siguientes:</p> <ul style="list-style-type: none"> • Soporte de resoluciones de hasta 1366x768 pixeles (WXGA). • Interfaz de usuario renovada. • Soporte para reproducción de video en formato WebM. • Tienda de música similar a iTunes.

<p>Ice Cream Sandwich (v4.0)</p>	<p>La llegada de Android 4.0 Ice Cream Sandwich el 19 de octubre de 2011 significó un importante paso en la evolución de Android que no solo vio renovada casi por completo su interfaz de usuario con el nuevo diseño Holo, sino que volvió a integrar el sistema operativo en sus versiones para Tablet y Smartphones. entre sus nuevas características podemos destacar las siguientes funciones:</p> <ul style="list-style-type: none"> • La pantalla de desbloqueo también ha sufrido una remodelación total • Desde la pantalla de desbloqueo podremos bajar la barra de notificaciones sin necesidad de desbloquear el terminal • barra deslizable que nos muestra un previo de las aplicaciones abiertas y de las últimas que hemos usado. • El corrector de texto en ICS ha sido rediseñado y mejorado. • En ICS, han añadido un gestor del tráfico de datos de internet • Una nueva barra de herramientas inteligente
<p>Jelly Bean (v4.1/v4.2/v4.3)</p>	<p>Éstas son las nueve principales características que aportará Android Jelly Bean :</p> <ul style="list-style-type: none"> • Optimización de las transiciones en la interfaz • Mejor pantalla de inicio con widgets e iconos cuyos tamaños se ajustarán automáticamente. • Mejor vista previa de las capturas fotográficas. • Mejora en la predicción de palabras del teclado y soporte offline para introducción de texto por voz. • Dieciocho nuevos idiomas y mejor accesibilidad, con soporte Braille. • Android Beam (tecnología NFC) para compartir archivos entre dispositivos con NFC. • Mejora de la barra de notificaciones, desde donde podremos devolver una llamada, leer el comienzo de un email, interactuar en redes sociales, etc. • Mejora en el reconocimiento de voz de Google. • Google Now.

<p>KitKat (v4.4)</p>	<p>Se trata de una actualización con un enorme número de nuevos detalles que se ha ganado el nombre propio de KitKat. Lo nuevo en esta versión es:</p> <ul style="list-style-type: none"> • Modo pantalla completa. • SMS y MMS en Hangouts. • Multitarea más rápida. • Impresión en nube. • Quickoffice integrado. • Rediseño de la aplicación de email por defecto (no la de Gmail). • Soporte para pagos NFC. • Fotografía HDR. • Sandbox de seguridad para aplicaciones. • Soporte para Chromecast. • Control de TVs con infrarrojos. • Gestión fácil de la pantalla home. • Reproducción de audio en bajo consumo. • Podómetro integrado. • Nuevos modos de localización (modo preciso o modo ahorro de batería).
<p>Android (5.0) Lollipop</p>	<p>El diseño de Google Now se expande, documenta y aplica a todo Android: llega Material Design, un soplo de aire fresco que ya se iba haciendo necesario tras seis años improvisando sin unas reglas claras. Material Design llegaba en varias aplicaciones de Google, pero ahora la pelota estaba en el tejado de los desarrolladores para que adaptaran sus aplicaciones a este nuevo diseño.</p>
<p>Android (6.0) Marshmallow</p>	<p>Llegamos a la sexta versión de Android, que vio la luz el 5 de octubre de 2015. Con Material Design ya asentado y siete años de recorrido, Google ya tiene a Android sobre raíles.</p> <p>Marshmallow introducía los permisos en tiempo de ejecución y Now On Tap. En Marshmallow el foco es seguir mejorando y cohesionando todo el sistema tras casi una década de desarrollo. Un aspecto que necesitaba una vuelta de tuerca era el sistema de permisos de todo o nada que existía hasta entonces. Con los permisos en tiempo de ejecución, las aplicaciones te pueden pedir permiso para usar cierta función (cámara, micrófono...) solo cuando lo necesitan.</p>

	<p>Otra preocupación cada vez mayor es la autonomía de la batería. Android avanza, el hardware avanza pero las baterías... no demasiado. Google se saca de la manga el Modo Doze, una especie de policía de la batería que obliga a las aplicaciones a dormir y reduce la velocidad de la CPU cuando la pantalla está apagada, para alargar la duración de la batería.</p> <p>El huevo de pascua de Marshmallow es básicamente el mismo de Lollipop, pero con marshmallow y "multiplayer".</p> <p>Llega el soporte para las tecnologías de moda: USB-C, modo 4K para aplicaciones, multiventana (experimental) y el soporte nativo para el lector de huellas. Uno que se pierde por el camino es el soporte para Miracast, que desaparece.</p> <p>Con Marshmallow llega Direct Share, la forma más rápida de enviar contenido a un contacto específico y Now On Tap, ese botón mágico que busca qué hay en tu pantalla para ofrecerte información relacionada.</p>
<p>Android (7.0) Nougat</p>	<p>Android Nougat siguió los pasos de Marshmallow refinando pequeños elementos heredados que necesitaban atención.</p> <p>Nougat traía la vista a pantalla partida y las respuestas rápidas. En cuanto al rendimiento, Nougat mejora mucho respecto a la versión de Marshmallow, haciéndolo efectivo incluso cuando el teléfono está en movimiento. Además, el nuevo compilador JIT reduce en un 75% la instalación de una aplicación y requiere de menos almacenamiento.</p> <p>Llegan la respuesta rápida, directamente desde la notificación de Android, la plataforma VR Daydream, el modo multiventana, el soporte Picture-in-Picture (solo en Android TV) y los gráficos de consola con Vulkan 3D.</p>
<p>Android (8.0) Oreo</p>	<p>Android Oreo veía la luz el 21 de agosto de 2017. Una vez más, Google debía poner algo de orden en un sistema cada vez más aquejado con la fragmentación. Llegaba así Project Treble, una buena promesa de actualizaciones más rápidas, al menos en teoría.</p> <p>En esta versión fue añadida la API de autocompletado de formularios y el modo PIP. Project Treble es la estrella de Android Oreo. Una nueva arquitectura modular del sistema para facilitar el proceso de actualizar un terminal y, teóricamente, lograr que lleven menos trabajo y, por tanto, te lleguen antes. Eso sí, probablemente no veremos su impacto hasta dentro de unos años.</p>

TABLA 3: EVOLUCIÓN DE LA PLATAFORMA ANDROID [9]

4.3 Arquitectura de Android

En las siguientes líneas se dará una visión global por capas de cuál es la arquitectura empleada en Android. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores.

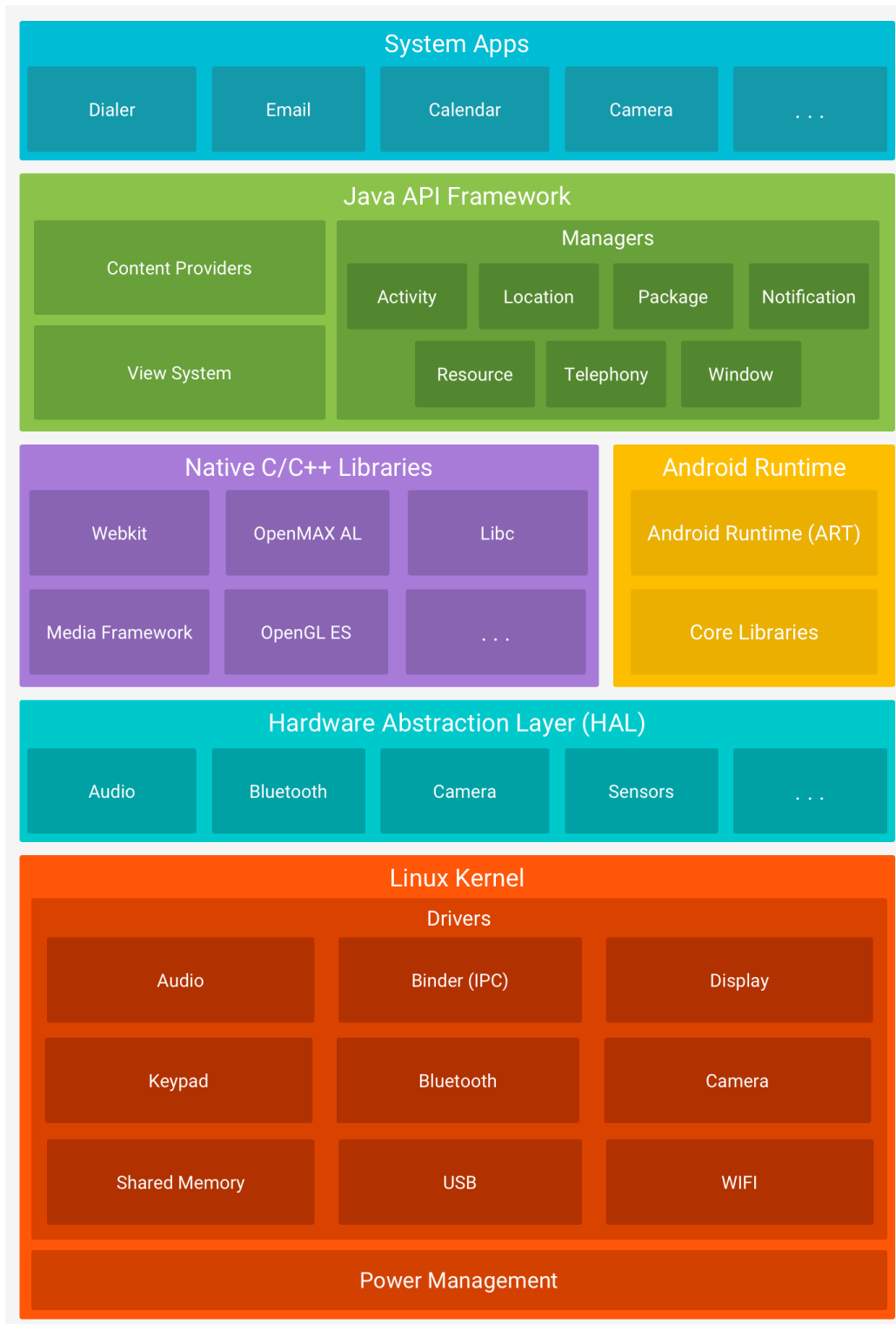


FIGURA 5: ARQUITECTURA DE ANDROID [10]

4.3.1 Nivel de aplicaciones

Android incluye una serie de aplicaciones básicas como un cliente de correo electrónico, un programa para mandar SMS, calendario, mapas, navegador web o contactos entre otros. Todas las aplicaciones están escritas en el lenguaje de programación Java y su desarrollo es posible gracias al Android SDK que provee de las herramientas e interfaces de programación necesarias.

4.3.2 Nivel del framework de aplicaciones

Android ofrece a los desarrolladores la capacidad de utilizar cualquier dispositivo presente en el teléfono para desarrollar una aplicación, pudiendo acceder por ejemplo a información de geolocalización a través del GPS ejecutar servicios en segundo plano, establecer alarmas, añadir notificaciones a la barra de estado, entre otras muchas cosas.

Este sistema está pensado para poder reutilizar de la forma más sencilla posible los componentes de cualquier aplicación y de esta forma cualquier aplicación puede hacer públicas sus características para que otras aplicaciones puedan utilizarlas (sujetas a posibles restricciones de seguridad).

En este nivel, situado entre las aplicaciones de usuario y las librerías del sistema, encontramos los siguientes servicios disponibles:

- **Gestor de Actividades:** se encarga de gestionar el ciclo de vida de las aplicaciones, así como de establecer un sistema de navegación entre las mismas.
- **Gestor de Ventanas:** servicio que se encarga de ofrecer una interfaz de acceso al gestor de ventanas del propio sistema operativo.
- **Proveedores de Contenido:** que permiten a una aplicación acceder a los datos de otras aplicaciones, como por ejemplo los datos de los contactos, y a su vez compartir los datos de la propia aplicación con el resto.
- **Sistema de Vistas:** Ofrece un gran número de vistas que pueden ser usadas para desarrollar las aplicaciones. Las vistas disponibles van desde las listas o las galerías pasando por formularios para introducir texto, botones o un navegador web.
- **Gestor de Paquetes:** permite obtener información relativa a las aplicaciones que están instaladas actualmente en el dispositivo.
- **Gestor de Telefonía:** provee de acceso a la información relativa a los servicios de telefonía del dispositivo como por ejemplo el estado de los mismos (si se está realizando una llamada o no, etc.).
- **Gestor de Recursos:** da acceso a los recursos usados por las aplicaciones como por ejemplo las imágenes, las cadenas de texto mostradas o los archivos XML en los que se especifica el diseño de la interfaz.
- **Gestor de Localización:** este componente permite por ejemplo que las aplicaciones obtengan información sobre la localización geográfica del dispositivo y puedan lanzar algún evento en función de esta información
- **Gestor de Notificaciones:** permite por ejemplo que una aplicación muestre una notificación en la barra de estado.
- **Servicio XMPP:** API que sirve para poder acceder a este servicio de intercambio de mensajes en formato XML.

4.3.3 Librerías

El conjunto de librerías de las que se compone Android es muy variado y extenso. Por un lado, encontramos las librerías base de Android, escritas en Java y que ofrecen un conjunto de funcionalidades comunes a cualquier sistema operativo que pueden ser utilizadas en tiempo de ejecución por las aplicaciones. Estas librerías se encuentran empaquetadas en el archivo “android.jar” que se distribuye junto al SDK de Android. Por otro lado, Android incluye un conjunto de librerías escritas en C o C++ que son usadas por los distintos componentes que lo integran y que además pueden ser utilizadas por las aplicaciones a través del framework de aplicaciones. A continuación, se describen las principales características de algunas de las librerías del sistema:

- **Librerías del sistema “Bionic libc”:** desarrolladas a partir de las librerías estándar de C creadas por el organismo BSD y modificadas especialmente para sistemas empujados basados en Linux.
- **Librerías multimedia:** basadas en las librerías de OpenCore soportan la reproducción y grabación de un gran número de formatos de audio y video, así como de imágenes, entre los que se incluyen MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
- **Librerías “Surface Manager”:** estas librerías, usadas por el gestor de ventanas del nivel superior de la arquitectura de Android, gestionan el acceso a los servicios del sistema relacionados con la composición de ventanas para las aplicaciones que estén activas en algún momento.
- **WebKit:** es un motor de renderizado utilizado en los navegadores web que ofrece una serie de características básicas que permite el visionado de las páginas web y otras funcionalidades como por ejemplo mantener un historial.
- **SKIA:** es la librería utilizada por Android para dibujar gráficos en dos dimensiones, texto o imágenes. Las siglas se corresponden con los términos en inglés Skia Graphics Library, siendo Skia la compañía encargada del desarrollo de esta librería escrita en C++ y distribuida bajo una licencia de código abierto.
- **Librerías 3D:** las cuales utilizan el API Open GL para sistemas empujados y permiten que los dispositivos muestren gráficos tridimensionales haciendo uso de hardware específico para la aceleración de gráficos 3D (cuando el terminal lo posea) o bien mediante software.
- **FreeType:** estas librerías son utilizadas como motor de renderizado de las distintas fuentes disponibles en el sistema.
- **SQLite:** motor de base de datos disponible para todas las aplicaciones que se caracteriza por ser ligero y muy competente, implementando la mayor parte del estándar SQL-92.
- **Librerías SSL:** posibilitan la utilización de este protocolo para establecer comunicaciones seguras.

4.3.4 Tiempo de ejecución de Android

Para los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, cada app ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android (ART). El ART está escrito para ejecutar varias máquinas virtuales en dispositivos de memoria baja ejecutando archivos DEX, un formato de código de bytes diseñado especialmente para

Android y optimizado para ocupar un espacio de memoria mínimo. Crea cadenas de herramientas, como Jack [11], y compila fuentes de Java en código de bytes DEX que se pueden ejecutar en la plataforma Android.

Estas son algunas de las funciones principales del ART:

- compilación ahead-of-time (AOT) y just-in-time (JIT);
- recolección de elementos no usados (GC) optimizada;
- mejor compatibilidad con la depuración, como un generador de perfiles de muestras dedicado, excepciones de diagnóstico detalladas e informes de fallos, y la capacidad de establecer puntos de control para controlar campos específicos.

Antes de Android 5.0 (nivel de API 21), Dalvik era el tiempo de ejecución del sistema operativo. Si tu app se ejecuta bien en el ART, también debe funcionar en Dalvik, pero es posible que no suceda lo contrario.

En Android también se incluye un conjunto de bibliotecas de tiempo de ejecución centrales que proporcionan la mayor parte de la funcionalidad del lenguaje de programación Java; se incluyen algunas funciones del lenguaje Java 8 [12], que el framework de la Java API usa.

4.3.5 Kernel Linux

Como se mencionó anteriormente, el “cerebro” de Android se basa en el kernel de Linux sirviendo como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura de Android.

A parte de esto, el kernel está encargado de gestionar los servicios de seguridad, la gestión de la memoria, de los procesos, del protocolo de red o los drivers del sistema, entre otras funciones.

4.4 Aplicaciones

Una de las características principales de Android es que cualquier aplicación puede utilizar componentes de otras, siempre y cuando cuente con los permisos adecuados. Por este motivo el sistema debe ser capaz de iniciar cualquier aplicación cuando sea preciso, y gracias a que en Android una aplicación no tiene un único punto de entrada, el sistema puede instanciar los componentes necesarios en cada momento.

Por defecto, cuando se inicia una aplicación, Android asigna a esta un proceso de Linux que cuenta con un único hilo de ejecución, por lo que todos sus componentes serán ejecutados en ese proceso e hilo. Sin embargo, se puede asignar un proceso diferente para cada componente, así como lanzar nuevas hebras de ejecución para cada proceso.

En el siguiente apartado serán descritos los cuatro tipos de componentes de los que puede constar una aplicación en Android, así como el mecanismo de activación de los mismos y su ciclo de vida.

4.4.1 Componentes de una Aplicación

Los componentes que componen una aplicación en Android son los siguientes:

Vista (View)

Las vistas son los elementos que componen la interfaz de usuario de una aplicación. por ejemplo, un botón, una entrada de texto, ... Todas las vistas van a ser objetos descendientes de la clase View y por tanto, pueden ser definidos utilizando código Java. Sin embargo, lo habitual va a ser definir las vistas utilizando un fichero XML y dejar que el sistema cree los objetos por nosotros a partir de este fichero. Esta forma de trabajar es muy similar a la definición de una página web utilizando código HTML.

Layout

Un Layout es un conjunto de vistas agrupadas de una determinada forma. Vamos a disponer de diferentes tipos de Layout para organizar las vistas de forma lineal, en cuadrícula o indicando la posición absoluta de cada vista. Los Layouts también son objetos descendientes de la clase VIEW. Igual que las vistas los Layouts pueden ser definidos en código, aunque la forma habitual de definirlos es utilizando código XML.

Actividad (Activity)

Una aplicación en Android va a estar formada por un conjunto de elementos básicos de visualización, coloquialmente conocidos como pantallas de la aplicación. En Android cada uno de estos elementos, o pantallas, se conoce como ACTIVIDAD. Su función principal es la creación del interfaz de usuario. Una aplicación suele necesitar varias ACTIVIDADES para crear el interfaz de usuario. Las diferentes ACTIVIDADES creadas serán independientes entre sí, aunque todas trabajarán para un objetivo común. Toda actividad ha de pertenecer a una clase descendiente de ACTIVITY.

Servicio (Service)

Un Servicio [13] es un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario. Otro componente

de la aplicación puede iniciar un servicio y continuará ejecutándose en segundo plano, aunque el usuario cambie a otra aplicación. Además, un componente puede enlazarse con un servicio para interactuar con él e incluso realizar una comunicación entre procesos (IPC). Por ejemplo, un servicio puede manejar transacciones de red, reproducir música, realizar I/O de archivos o interactuar con un proveedor de contenido, todo en segundo plano.

Intención (Intent)

Una INTENCIÓN representa la voluntad de realizar alguna acción; como realizar una llamada de teléfono, visualizar una página web. Se utiliza cada vez que queramos:

- lanzar una ACTIVIDAD
- lanzar un SERVICIO
- enviar un ANUNCIO DE TIPO BROADCAST
- comunicarnos con un SERVICIO

Los componentes lanzados pueden ser internos o externos a nuestra aplicación. También utilizaremos las INTENCIONES para el intercambio de información entre estos componentes.

Receptor de anuncios (Broadcast receiver)

Un RECEPTOR DE ANUNCIOS recibe y reacciona ante anuncios de tipo broadcast. Los anuncios BROADCAST pueden ser originados por el sistema o por las aplicaciones. Algunos tipos de anuncios originados por el sistema son: BATERÍA BAJA, LLAMADA ENTRANTE, ... Las aplicaciones también pueden crear y lanzar nuevos tipos de ANUNCIOS BROADCAST. Los receptores de anuncios no disponen de interfaz de usuario, aunque pueden iniciar una actividad si lo estiman oportuno.

Proveedores de Contenido (Content Provider)

En muchas ocasiones las aplicaciones instaladas en un terminal Android necesitan compartir información. Android define un mecanismo estándar para que las aplicaciones puedan compartir datos sin necesidad de comprometer la seguridad del sistema de ficheros. Con este mecanismo podremos acceder a datos de otras aplicaciones, como la lista de contactos, o proporcionar datos a otras aplicaciones.

4.4.2 Ciclo de vida de una aplicación Android

En esta sección se comentarán los principales aspectos del ciclo de vida de los componentes de los que puede constar una aplicación.

Actividad

Una Actividad es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción, como marcar un número telefónico, tomar una foto, enviar un correo electrónico o ver un mapa. A cada actividad se le asigna una ventana en la que se puede dibujar su interfaz de usuario. La ventana generalmente abarca toda la pantalla, pero en ocasiones puede ser más pequeña que esta y quedar "flotando" encima de otras ventanas.

Una aplicación generalmente consiste en múltiples actividades vinculadas de forma flexible entre sí. Normalmente, una actividad en una aplicación se especifica como la actividad "principal" que se presenta al usuario cuando este inicia la aplicación por primera vez. Cada actividad puede a su vez iniciar otra actividad para poder realizar diferentes acciones. Cada vez que se inicia una actividad nueva, se detiene la actividad anterior, pero el sistema conserva la actividad en una pila (la "pila de actividades"). Cuando se inicia una actividad nueva, se la incluye en la pila de actividades y capta el foco del usuario. La pila de actividades cumple con el mecanismo de pila "el último en entrar es el primero en salir", por lo que, cuando el usuario termina de interactuar con la actividad actual y presiona el botón *Atrás*, se quita de la pila (y se destruye) y se reanuda la actividad anterior.

Cuando se detiene una actividad porque se inicia otra, se notifica el cambio de estado a través de los métodos callback del ciclo de vida de la actividad. Existen diferentes métodos callback que podría recibir una actividad como consecuencia de un cambio de estado (ya sea que el sistema la esté creando, deteniendo, reanudando o destruyendo) y cada callback te da la oportunidad de realizar una tarea específica que resulta adecuada para ese cambio de estado. Por ejemplo, cuando se detiene una actividad, esta debería liberar los objetos grandes, como una conexión de red o a la base de datos. Cuando se reanuda la actividad, puedes volver a adquirir los recursos necesarios y reanudar las acciones que se interrumpieron. Todas estas transiciones de estado forman parte del ciclo de vida de la actividad.

El resto de este documento aborda los conceptos básicos con respecto a cómo crear y utilizar una actividad, incluido un artículo completo sobre cómo funciona el ciclo de vida de la actividad, de modo que puedas administrar correctamente la transición entre diferentes estados de la misma [14].

En conjunto, estos métodos definen el ciclo de vida completo de una actividad, en el cual podemos diferenciar tres bucles de ejecución:

- **Ciclo de vida completo** de una actividad que sucede entre la primera llamada a `onCreate` y la llamada final a `onDestroy`. La actividad lleva a cabo su configuración inicial en la llamada a `onCreate`, y libera todos los recursos que quedan en `onDestroy`.
- **Tiempo de vida visible** de una actividad que ocurre entre una llamada a `onStart` hasta que se produce una llamada a `onStop`. Durante este tiempo, el usuario puede ver la actividad en pantalla, aunque puede que no sea en primer plano (interactuando con el usuario). Entre estos dos métodos, se pueden mantener los recursos que se necesitan para mostrar la actividad para el usuario.
- **Tiempo de vida en primer plano** de una actividad que ocurre entre una llamada a `onResume` hasta que se produce una llamada a `onPause`. Durante este tiempo, la actividad se encuentra en primer plano y el usuario está interactuando con la misma.

El siguiente diagrama ilustra estos bucles y los caminos que una actividad puede tomar entre los distintos estados. Los óvalos de color son los estados más importantes en los que se puede encontrar la actividad. Los rectángulos representan los métodos que se puede implementar para realizar operaciones cuando la actividad transita entre estados

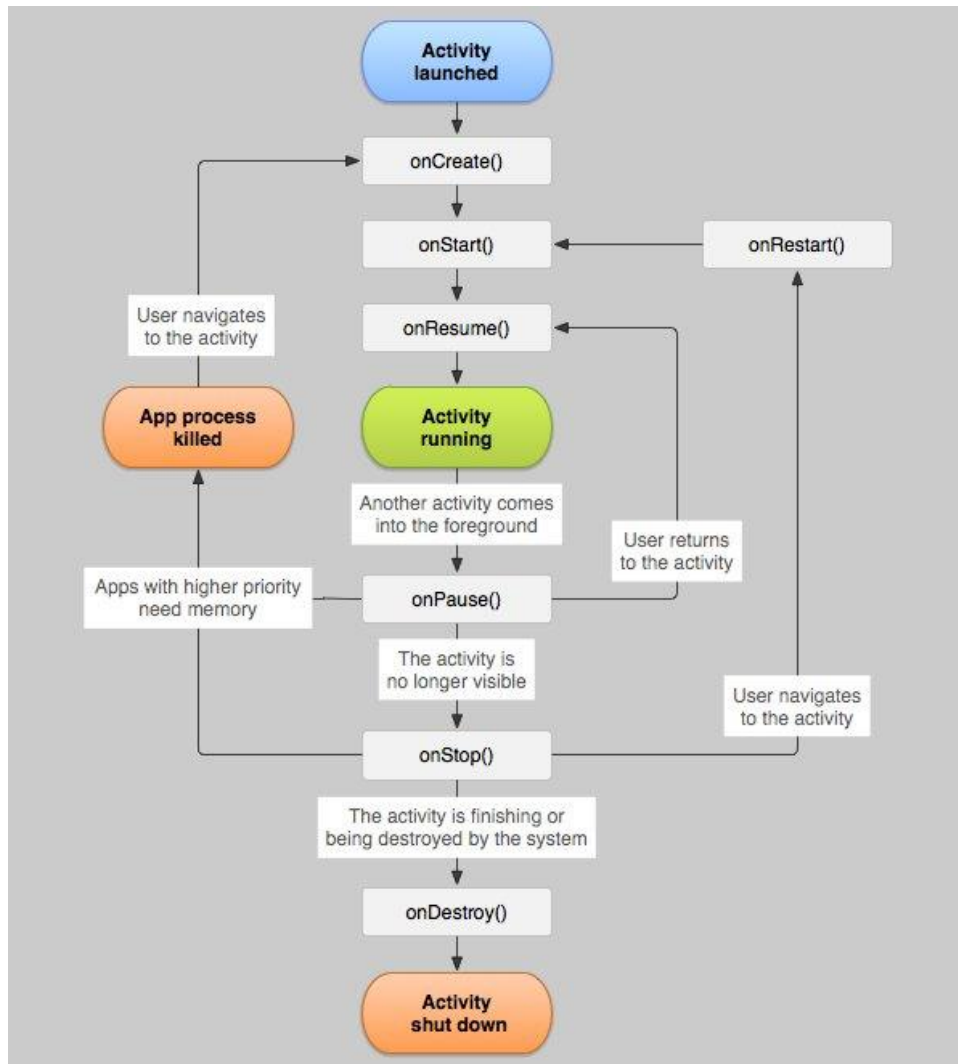


FIGURA 6: CICLO DE VIDA DE UNA ACTIVIDAD

Como se mencionó en esta sección cuando el sistema finaliza una actividad se debe tener en cuenta que él, cuando el usuario quiera volver a interactuar con dicha actividad esta debe presentarse tal y como se encontraba antes de ser expulsada de memoria por el sistema.

Para ello Android ofrece métodos dentro de su API que guardan el estado de la aplicación antes de que pueda ser eliminada (concretamente antes de la llamada a onPause) y recuperan el mismo cuando la actividad vuelve a tener el foco principal, aparte de objetos como Bundle con el que el estado de la aplicación es guardado como pares de nombre-valor.

- **Servicio**

Se puede acceder a un servicio de dos formas: iniciándolo y dejándolo ejecutar en segundo plano hasta que alguien lo pare o hasta que el mismo servicio finalice su ejecución con las funciones startService y stopService y, en segundo lugar, puede ser accedido programáticamente usando una **interfaz definida** y exportada por el servicio, utilizando los métodos bindService y unbindService.

Al igual que las actividades, los servicios tienen unos métodos que definen su ciclo de vida, con los que se puede monitorizar las transiciones de un estado a otro, que son: onCreate,

onStart, y onDestroy. De esta manera se definen dos bucles de ejecución para estos componentes

- **Ciclo de vida completo:** queda definido entre la primera llamada que se hace a onCreate y el instante en que finaliza la llamada a onDestroy.
- **Ciclo de vida activo:** comienza con la llamada a onStart.

Mientras que onCreate y onDestroy son llamadas comunes a los servicios independientemente de cómo sean iniciados, onStart solo puede ser invocado por los servicios que fueron iniciados por startService. Así mismo si un servicio permite a otros componentes que accedan a él existen otros métodos que definen el estado de un servicio, y son: onBind, onUnbind y onBind.

El siguiente diagrama muestra las transiciones entre estados de un servicio. Aunque en el diagrama se separan los servicios que son creados con una llamada a startService de los que son creados con una llamada a bindService, se debe tener en cuenta que un servicio puede permitir a otros componentes asociarse con él, por lo que cualquier componente de este tipo puede recibir llamadas onBind y onUnbind.

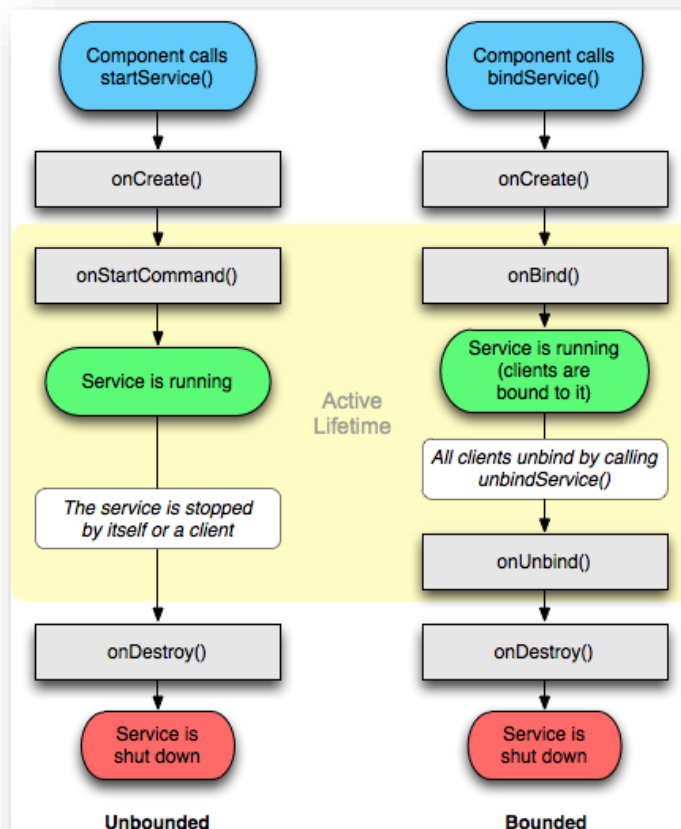


Figura 7: BUCLE DE EJECUCIÓN DE SERVICIO

- **Receptores de difusión** Estos componentes tienen un solo método asociado con el ciclo de vida que es onReceive. Cuando llega un mensaje a un receptor de difusión se llama a este método pasándole dicho mensaje. En este momento, y sólo mientras ejecuta este método, el receptor se considera que está activo y por tanto un proceso

que contenga a uno de estos componentes activos no podrá ser eliminado de la memoria del sistema.

- **Procesos** A la hora de determinar qué proceso debe ser eliminado para ceder sus recursos a otro, Android sitúa cada proceso dentro de una jerarquía de importancia basada en el número de componentes que están ejecutándose y el estado de los mismos, siendo eliminados primero los procesos con menor

prioridad en la jerarquía para después ir suprimiendo los procesos de mayor prioridad si es necesario

Esta jerarquía consiste en cinco niveles ordenados a continuación de mayor a menor importancia:

Primer plano: aquellos procesos que son requeridos por la actividad del usuario. Están en este nivel de la jerarquía los procesos que cumplen una de las siguientes condiciones:

- Están ejecutando una actividad con la que el usuario esta interactuando
- Contiene un servicio con el que el usuario esta interactuando.
- Contiene un servicio que está ejecutando alguna de sus llamadas de ciclo de vida (onCreate, onStart, onDestroy)
- Contiene un receptor de difusión que está ejecutando el método onReceive.

En un mismo instante existen pocos procesos de este nivel de la jerarquía en memoria, los cuales son eliminados como último recurso cuando el sistema se ve obligado a realizar un re-paginación de la memoria, por lo que la eliminación de estos procesos ayuda a que la interfaz responda a las acciones del usuario de forma apropiada.

Visibles: son los procesos que no tienen componentes en primer plano pero que aún son visibles por el usuario. Cumplen alguna de estas condiciones:

- Contiene una actividad que no está en primer plano, pero aún es visible por el usuario (se ha invocado el método onPause)
- Contiene un servicio que está enlazado a una actividad que es visible

Estos procesos no son eliminados a no ser que sea necesario para que los procesos de primer plano sigan ejecutando.

Servicios: son los procesos que se encuentran ejecutando un componente de este tipo, pero no se ajustan a las prioridades superiores. No son visibles para el usuario, aunque suelen llevar a cabo tareas importantes para este (como reproducir música), por esto el sistema los mantiene en memoria hasta que es necesario que sean eliminados debido a que los procesos en primer plano o los visibles necesitan más memoria.

Segundo plano: son los procesos que contienen actividades que no son visibles para el usuario (el método onStop ha sido invocado). Se pueden eliminar en cualquier momento de forma que los procesos de los niveles superiores de la jerarquía puedan obtener la memoria utilizada por los mismos. Normalmente hay varios procesos en segundo plano por lo que se mantiene una lista con los que contienen actividades que han sido usadas recientemente eliminándose por tanto en primer lugar los procesos más antiguos.

Vacíos: son los procesos que no contienen ningún componente activo y que únicamente se mantienen en memoria como cache para futuras ejecuciones de forma que se mejore la rapidez con la que una aplicación inicia.

Android asigna siempre un proceso al nivel más elevado posible de prioridad, teniendo en cuenta ciertas consideraciones para ello, como por ejemplo la importancia de los componentes activos (en el caso de que un proceso contenga un servicio y una actividad visible, el proceso será asignado al nivel visible). Por otro lado, la importancia de un proceso puede verse incrementada debido a que otros dependan del mismo, como por ejemplo en el caso de que un proceso este sirviendo a otro, el servidor nunca debe ser asignado a un nivel de importancia menor que el cliente.

Por último, en función de lo comentado a cerca del ciclo de vida de los procesos, es recomendable pensar que tipo de componente debe ejecutar cierta acción dentro de una aplicación. Por ejemplo, se debería asignar un servicio a una operación de descarga, la cual se supone va a consumir bastante tiempo, en lugar de una actividad para poder tener garantías de que la operación termina antes de que el proceso sea expulsado de la memoria.

4.5 Interfaz de Usuario Android

La plataforma Android dispone de dos clases Java especialmente diseñadas para la construcción de la interfaz de usuario, o UI, de una aplicación. Estas clases son View y ViewGroup.

La clase View funciona como unidad base para la creación de las UI y para implementar otros componentes de UI como son los widgets. Estos últimos son objetos predeterminados para crear botones, campos de texto, entre otras cosas. Por otra parte, la clase ViewGroup funciona como base para la implementación de Layouts, los cuales ofrecen diferentes tipos de diseño que establecen la disposición de los elementos View en la UI. Estos pueden ser lineales, absolutos, relativos, y mucho más.

Un objeto View puede ser definido como una estructura de datos que establece los parámetros de diseño y el contenido de un área rectangular específica dentro de la pantalla del dispositivo. Estos objetos también manejan el tamaño, diseño, enfoque y desplazamiento del área rectangular en la cual residen.

Tal y como se muestra en la **figura 7**, la UI de una actividad es esquematizada a través de un árbol jerárquico de nodos View y ViewGroup [15]. Este árbol puede ser tan simple o complejo como el diseño de la aplicación lo exija.

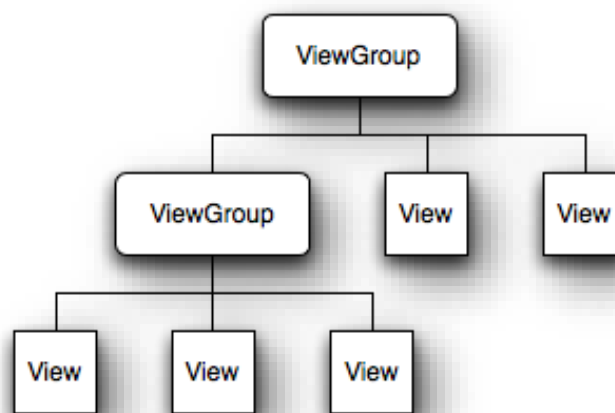


Figura 8: ÁRBOL JERÁRQUICO DE UNA UI EN ANDROID

Para poder representar este árbol jerárquico en la pantalla del dispositivo, la actividad únicamente debe hacer una llamada a un método específico contenido en la clase View y pasar una referencia al objeto ubicado en la raíz. Luego el sistema utiliza esta referencia y los parámetros contenidos en el diseño del Layout para hacer las mediciones y dibujar el árbol. Para ello, el nodo raíz solicita a sus nodos hijos que se dibujen a ellos mismo, a la vez que los nodos ViewGroup llaman a sus nodos hijos para que también lo hagan. Cada uno de los nodos solicitan un tamaño y una posición dentro de sus padres, quienes tienen la última decisión de qué tamaño pueden tener y en donde se pueden ubicar. Como los elementos son dibujados

de manera jerárquica, es decir, desde la raíz hasta las hojas, si existen elementos solapados prevalecerá el último elemento que haya sido dibujado.

Una de las principales características de implementación de la plataforma Android, es la posibilidad de declarar la arquitectura del Layout a través de un archivo XML, el cual posee una estructura y codificación simples que permite ser leída por humanos. Los elementos de estos archivos pueden ser de tipo ViewGroup o View. En la **figura 8** se puede observar un Layout lineal (ViewGroup) con orientación vertical que contiene botón (View) y un elemento de texto (View). Este podría ser más complejo y contener a su vez otro tipo de elemento ViewGroup si así lo requiere la actividad

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

FIGURA 9: ARCHIVO XML DE UNA ACTIVIDAD EN ANDROID

Cada tipo de elemento XML en el archivo del layout tiene su representación de clase Java. El elemento <TextView> tiene una clase Java llamada TextView.

Como se había mencionado anteriormente, existe una subclase de objetos View denominados widgets que sirven de interfaz para la interacción con el usuario. La plataforma Android posee un grupo de widgets como botones, casillas de selección y campos de texto, ya implementados y listos para ser agregados de manera rápida en la UI. También existen otros widgets más complejos como relojes y controles de zoom. Estos widgets pueden ser personalizados, combinados o simplemente creados desde cero.

Una vez establecido el diseño y componentes de una UI, es posible implementar acciones que respondan a interacciones con el usuario e iniciar intents. Para notificarle esto a la aplicación, la plataforma posee diversas clases y métodos que le facilitan la labor al desarrollador.

4.6 Recursos

Una buena práctica de programación en Android consiste en referenciar los recursos (imágenes, textos, etc.) de una aplicación de forma externa a la misma, de esta manera el mantenimiento de estos recursos es totalmente independiente a la aplicación. Otra ventaja de realizar esta práctica es que se pueden definir recursos alternativos para distintas configuraciones de dispositivo como el lenguaje, tamaño de pantalla, etc.

Para ello, por un lado, es necesario organizar los recursos empleados por la aplicación en distintos directorios bajo el directorio “res” que es el directorio raíz asignado a estos elementos. De esta forma, las imágenes se almacenan en un directorio distinto al de las cadenas de texto o de los ficheros que contienen el layout de una aplicación. Por otro lado, cuando se desea definir recursos alternativos, para por ejemplo una imagen, se debe crear un directorio que identifique la configuración alternativa con la que el recurso va a ser utilizado [16].

4.7 Proveedores de contenido

Los proveedores de contenido almacenan y recuperan datos haciéndolos accesibles a todas las aplicaciones. Son la única manera de compartir datos entre aplicaciones, pues no hay un área de almacenamiento común al que todos los paquetes de Android pueden acceder.

Android incluye una serie de proveedores de contenido para los tipos de datos comunes como audio, vídeo, imágenes, información de contacto personal, etc.; siendo necesario para algunos de ellos tener el permiso adecuado para leer los datos que almacena, añadiendo dicho permiso en el archivo `AndroidManifest` de la aplicación.

Por tanto, para compartir datos entre varias aplicaciones, se presentan dos opciones que involucran el uso de un proveedor de contenido: crear uno propio o agregar los datos a uno existente, siempre que el tipo de datos que se quiere almacenar sea compatible.

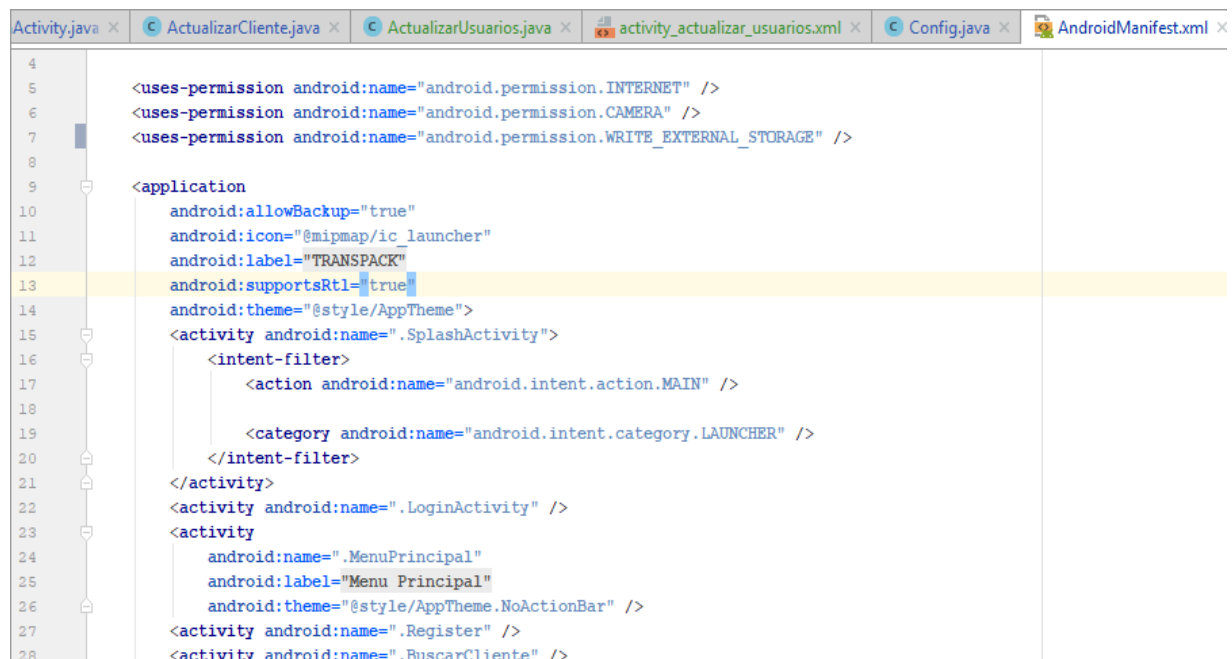
Para acceder a los datos que almacena un proveedor de contenido de forma independiente de cómo se almacenan estos, se define una interfaz de acceso común a cualquier proveedor a través de un objeto del API de Android llamado `ContentResolver` que ofrece una serie de métodos para interactuar con los datos almacenados. Cuando se realiza una consulta de los datos almacenados por un proveedor de contenido, el `ContentResolver` utiliza un objeto `Cursor` que permite navegar por la estructura tabular en la que son almacenados los datos, similar a como se almacenan en una base de datos.

4.8 El archivo `AndroidManifest.xml`

La base para cualquier aplicación Android es el archivo `AndroidManifest.xml` que se encuentra en la raíz de directorios de cualquier trabajo. Dentro de este archivo se declara todo lo que se encuentre dentro de la aplicación, las actividades, los servicios, etc. Cómo todas estas piezas interactúan entre sí y con el sistema, podemos por ejemplo indicar qué actividad (o actividades) deben aparecer en el menú principal del dispositivo, también conocido como el launcher. Cada vez que creamos un nuevo trabajo Android, obtendremos un archivo `manifest` inicial que se genera automáticamente, y que abarca los elementos básicos de una aplicación sencilla; la declaración de una actividad y nada más. Conforme aumente la complejidad de

las funciones que queremos que realicen nuestras aplicaciones, este archivo irá creciendo en líneas de código y en declaraciones de elementos distintos.

La siguiente figura muestra la estructura general del archivo AndroidManifest.xml y los elementos que puede contener, los cuales son listados a continuación, no permitiéndose definir otros que no aparezcan en esta lista: <action>, <activity>, <activity-alias>, <application>, <category>, <data>, <grant-uri-permission>, <instrumentation>, <intent-filter>, <manifest>, <meta-data>, <permission>, <permission-group>, <permission-tree>, <provider>, <receiver>, <service>, <supports-screens>, <uses-configuration>, <uses-feature>, <uses-library>, <uses-permission> y <uses-sdk>.[17]



```
4
5 <uses-permission android:name="android.permission.INTERNET" />
6 <uses-permission android:name="android.permission.CAMERA" />
7 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
8
9 <application
10     android:allowBackup="true"
11     android:icon="@mipmap/ic_launcher"
12     android:label="TRANSPACK"
13     android:supportsRtl="true"
14     android:theme="@style/AppTheme">
15     <activity android:name=".SplashActivity">
16         <intent-filter>
17             <action android:name="android.intent.action.MAIN" />
18
19             <category android:name="android.intent.category.LAUNCHER" />
20         </intent-filter>
21     </activity>
22     <activity android:name=".LoginActivity" />
23     <activity
24         android:name=".MenuPrincipal"
25         android:label="Menu Principal"
26         android:theme="@style/AppTheme.NoActionBar" />
27     <activity android:name=".Register" />
28     <activity android:name=".BuscarCliente" />
```

FIGURA 10: EL ARCHIVO ANDROIDMANIFEST.XML

4.9 Permisos

Un punto clave del diseño de la arquitectura de seguridad de Android es que, por defecto, ninguna aplicación tiene permisos para realizar operaciones que tengan un impacto negativo en otras aplicaciones, el sistema operativo, o el usuario. Esto incluye la lectura o escritura de datos privados del usuario (por ejemplo, contactos o mensajes de correo electrónico), leer o escribir archivos de otra aplicación, el acceso a la red, etc.

Por tanto, para poder utilizar ciertas características de un dispositivo se deben incluir en el archivo AndroidManifest.xml una o varias etiquetas <usespermission> en la que se identifica el permiso que la aplicación necesita. Por ejemplo, en la siguiente imagen se muestra parte del archivo de manifiesto en el que se indican los permisos que la aplicación necesita (como acceso a internet o al dispositivo de almacenamiento externo) y que cuando se proceda a

instalar la aplicación, será el instalador de paquetes quien otorgue los permisos solicitados por la misma.

```
android:versionName="1.0.0" />
<!-- This is the platform API where NativeActivity was introduced. -->
<uses-sdk android:minSdkVersion="9" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.READ_CALENDAR" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<application android:persistent="False" android:restoreAnyVersion="False" android:label="
android:debuggable="False" android:isSystem="False" android:isSystem="False" />
```

FIGURA 11: PERMISOS DENTRO DEL MANIFEST

Al igual que se definen permisos de más alto nivel como los indicados en el párrafo anterior, también se pueden definir permisos específicos para un componente determinado o incluso definir permisos personalizados.

5. Otras tecnologías usadas en el trabajo

5.1 Cliente-servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema [18].

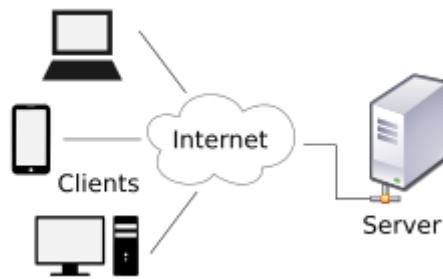


FIGURA 12: FIGURA DESCRIPTIVA DE LA ARQUITECTURA CLIENTE-SERVIDOR

5.2 El protocolo HTTP

Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador web o un spider) se lo conoce como "user agent" (agente del usuario). A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

5.3 El lenguaje de programación Java al lado del cliente

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con más de 10 millones de usuarios reportados.

5.4 MYSQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia. En la siguiente figura se muestra la interacción entre las diferentes tecnologías usadas en este trabajo.

5.5 Ubuntu server

Ubuntu Server es la versión para servidores de Ubuntu, el sistema operativo de Canonical basado en Linux. Canonical, compañía británica propiedad del empresario sudafricano Mark Shuttleworth, ofrece el sistema de manera gratuita y se financia por medio de servicios vinculados al sistema operativo y vendiendo soporte técnico. Además, al mantenerlo libre y gratuito, la empresa es capaz de aprovechar los desarrolladores de la comunidad para mejorar los componentes de su sistema operativo.

Cada seis meses se publica una nueva versión de Ubuntu la cual recibe soporte por parte de Canonical, durante dieciocho meses, por medio de actualizaciones de seguridad, parches para errores críticos y actualizaciones menores de programas. Las versiones LTS (Long Term Support o Soporte de Larga Duración), que se liberan cada dos años, reciben soporte durante cinco años en los sistemas de escritorio y de servidor.

5.6 LAMP

Es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

- **Linux**, el sistema operativo; En algunos casos también se refiere a LDAP.
- **Apache**, el servidor web;
- **MySQL/MariaDB**, el gestor de bases de datos;
- **Perl, PHP, o Python**, los lenguajes de programación.

La combinación de estas tecnologías es usada principalmente para definir la infraestructura de un servidor web, utilizando un paradigma de programación para el desarrollo.

A pesar de que el origen de estos programas de código abierto no fue específicamente diseñado para trabajar entre sí, la combinación se popularizó debido a su bajo coste de

adquisición y ubicuidad de sus componentes (ya que vienen pre-instalados en la mayoría de las distribuciones linux). Cuando son combinados, representan un conjunto de soluciones que proporcionan servidores de aplicaciones [19].

5.7 Webmin

Es una herramienta de configuración de sistemas accesible vía web para sistemas Unix, como GNU/Linux y OpenSolaris. Con él se pueden configurar aspectos internos de muchos sistemas operativos, como usuarios, cuotas de espacio, servicios, archivos de configuración, apagado del equipo, etcétera, así como modificar y controlar muchas aplicaciones libres, como el servidor web Apache, PHP, MySQL, DNS, Samba, DHCP, entre otros.

Está escrito en Perl, versión 5, ejecutándose como su propio proceso y servidor web. Por defecto se comunica mediante TCP a través del puerto 10000, y puede ser configurado para usar SSL si OpenSSL está instalado con módulos de Perl adicionales requeridos.

Está construido a partir de módulos, los cuales tienen una interfaz a los archivos de configuración y el servidor Webmin. Esto hace fácil la adición de nuevas funcionalidades sin mucho esfuerzo. Debido a su diseño modular, es posible para cualquier interesado escribir extensiones para configuración de escritorio.

Permite controlar varias máquinas a través de una interfaz simple, o iniciar sesión en otros servidores webmin de la misma subred o red de área local [20].

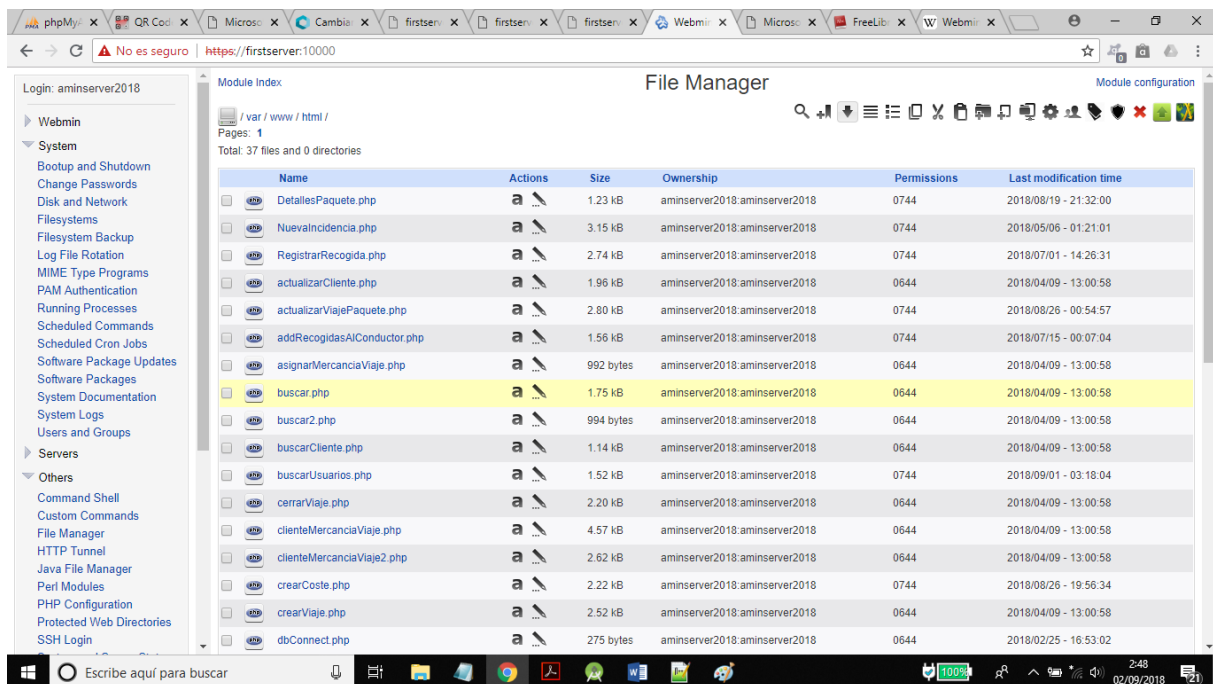


FIGURA 13: LA INTERFAZ DE WEBMIN

5.8 PHPMyAdmin

Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 72 idiomas. Se encuentra disponible bajo la licencia GPL Versión 2 [21].

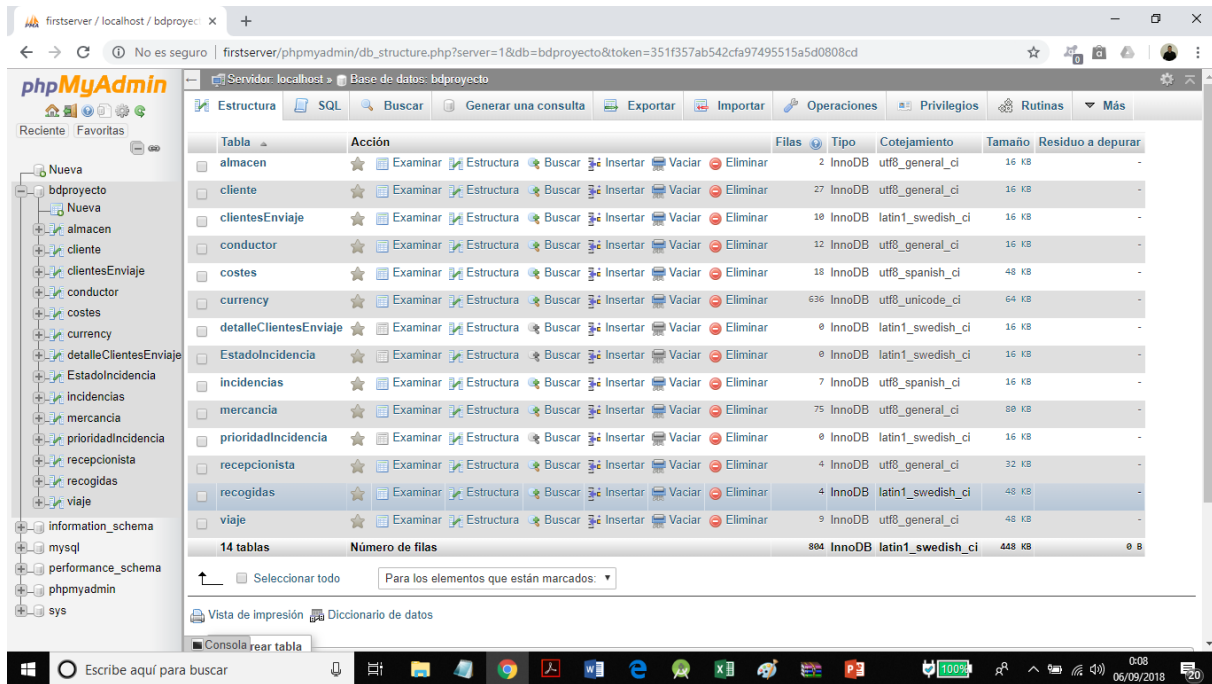


FIGURA 14:PHPMYADMIN

5.9 JSON

JSON (acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función `eval()`, lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

En la práctica, los argumentos a favor de la facilidad de desarrollo de analizadores o de sus rendimientos son poco relevantes, debido a las cuestiones de seguridad que plantea el uso de `eval()` y el auge del procesamiento nativo de XML incorporado en los navegadores modernos. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo, Google, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el no disponer de procesamiento XSLT para manipular los datos en el cliente.

Si bien es frecuente ver JSON posicionado *contra* XML, también es frecuente el uso de JSON y XML en la misma aplicación. Por ejemplo, una aplicación de cliente que integra datos

de Google Maps con datos meteorológicos en SOAP hacen necesario soportar ambos formatos.

6. Análisis, diseño e implementación de la solución

En este sexto capítulo del trabajo fin de grado se incluye una descripción del problema y un estudio de las alternativas. Es el más importante ya que describe todo el proceso, desde la primera idea hasta su puesta en marcha en el dispositivo.

6.1 Análisis

La idea de la aplicación está basada en una empresa familiar de transporte de mercancía que hace viajes desde Alemania hasta Marruecos y viceversa, el cliente entrega su paquete en la oficina en Alemania o las oficinas en Marruecos y la empresa se responsabiliza de llevarlos hasta el destino final y durante el viaje también se recogen mercancías y paquetes de otros clientes.

Como el jefe de la empresa es un transportista más la solución fue crear la aplicación en un dispositivo móvil para facilitar la tarea de reparto y recogida mejorando la gestión del viaje ahorrando tiempo y costo.

La aplicación tiene tres tipos de usuario:

El transportista: es el responsable de crear un viaje repartir la mercancía o paquetes relacionados con este viaje y al mismo tiempo recoger la mercancía de otros clientes registrándolos y registrando la mercancía.

El jefe: como he mencionado anteriormente es un transportista más, pero la diferencia es que el jefe puede gestionar y ver los detalles de todos los viajes activos mientras el simple transportista solo puede gestionar y ver datos relacionados con su viaje.

El recepcionista: Es el responsable de recibir las mercancías y paquetes, registrar al cliente en el caso que no sea registrado y registrar la mercancía.

La aplicación fue diseñada de forma que se puede adaptar (añadiendo y modificando funcionalidades según las necesidades) fácilmente a cualquier otra empresa de transporte de mercancías de tamaño pequeño o medio.

6.2 Requisitos de usuario

Una vez analizados los objetivos e idea inicial, se realiza la extracción de los requisitos de usuario.

Con los requisitos de usuario, el mismo indica una serie de restricciones o funcionalidades que debe cumplir la aplicación. Los tipos de requisitos de usuario son de capacidades (requeridas por los usuarios para resolver un problema o determinar un objetivo) y de restricciones (impuestas por los usuarios sobre la forma de cómo debe ser resuelto el problema o lograr el objetivo).

Los requisitos de usuario están definidos por los siguientes atributos:

Identificador: identifica de forma univoca un requisito, siguiendo las reglas de nombrado siguientes: CAP-X, para los requisitos de capacidad, y RES-X, para los requisitos de restricción. El valor de X es numérico empezando en uno.

Descripción: Especificación detallada del requisito.

Necesidad: Establece la necesidad del requisito dentro del trabajo, toma un valor dentro de la escala 1-4, siendo 4 la necesidad más alta y 1 la más baja.

Prioridad: Establece la prioridad del requisito dentro del trabajo, toma un valor dentro de la escala 1-4, siendo 4 la prioridad más alta y 1 la más baja.

Estabilidad: Establece la sensibilidad del requisito a ser modificado, toma los valores de “Estable” o “No Estable”.

6.2.1 Requisitos de capacidad

A continuación, se muestran los requisitos de usuario de capacidad:

Identificador	RUC-1: Navegar por Actividades
Descripción	El usuario será capaz de navegar por las Actividades permitidas depende de su perfil en la aplicación.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 4: REQUISITOS DE CAPACIDAD (1)

Identificador	RUC-2: Idioma
Descripción	La aplicación se podrá mostrar en inglés y español. Depende del idioma instalado en el smartphone.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 5: REQUISITOS DE CAPACIDAD (2)

Identificador	RUC- 3: El uso de la aplicación por varios usuarios
Descripción	El servidor será capaz de atender a varios usuarios a la vez.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 6: REQUISITOS DE CAPACIDAD (3)

Identificador	RUC-4: Mensajes de información y advertencia
Descripción	Se deben mostrar mensajes de información/ayuda cuando El usuario no introduce bien los datos de entrada, cuando ocurren todos los casos de errores
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 7: REQUISITOS DE CAPACIDAD (4)

Identificador	RUC-5: Menú de opciones
Descripción	Es necesario tener un menú de opciones para acceder a las diferentes pantallas de la aplicación.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 8: REQUISITOS DE CAPACIDAD (5)

Identificador	RUC-6: icono de la aplicación
Descripción	Para identificar mi aplicación de las otras instaladas en el móvil.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 9: REQUISITOS DE CAPACIDAD (6)

Identificador	RUC-7: Plataforma Android
Descripción	La aplicación será compatible con la plataforma Android.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 10: REQUISITOS DE CAPACIDAD (7)

Identificador	RUC- 8: datos de producción en tiempo real
Descripción	La aplicación tiene que ser capaz de mostrar los datos de cada viaje en tiempo real.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 11: REQUISITOS DE CAPACIDAD (8)

Identificador	RUC- 9: Estado de los paquetes en reparto
Descripción	La aplicación tiene que ser capaz de mostrar el estado de cada mercancía (entregada, pendiente de entregar...) en cualquier tiempo que el usuario lo solicita durante el reparto.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 12: REQUISITOS DE CAPACIDAD (9)

Identificador	RUC- 10: Identificación de los paquetes en el reparto
Descripción	La aplicación tiene que ser capaz de identificar la mercancía y su dueño mediante la lectura del código QR.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 13: REQUISITOS DE CAPACIDAD (10)

Identificador	RUC- 11: Generación del código Qr para cada mercancía
Descripción	La aplicación tiene que ser capaz de generar un código QR para cada para identificar la mercancía en el reparto
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 14: REQUISITOS DE CAPACIDAD (11)

Identificador	RUC- 12: Registro de costes de cada viaje
Descripción	La aplicación tiene que ser capaz registrar todos los costes de cada viaje.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 15: REQUISITOS DE CAPACIDAD (12)

Identificador	RUC- 13: Gestión de incidencias
Descripción	La aplicación tiene que ser capaz de gestionar las incidencias producidas en todos los viajes.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 16: REQUISITOS DE CAPACIDAD (13)

Identificador	RUC- 14: Pasar los paquetes de un viaje a otro
Descripción	La aplicación tiene que ser capaz de pasar un paquete de un viaje a otro.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 17: REQUISITOS DE CAPACIDAD (14)

Identificador	RUC- 15: Entregar los paquetes a los clientes
Descripción	La aplicación tiene que ser capaz de gestionar las entregas de los paquetes a los clientes.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 18: REQUISITOS DE CAPACIDAD (15)

Identificador	RUC- 16: Modificación del transportista durante el viaje
Descripción	La aplicación tiene que ser capaz de gestionar la modificación del transportista del viaje por otro.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 19: REQUISITOS DE CAPACIDAD (16)

Identificador	RUC- 17: Mostrar los detalles de los viajes activos
Descripción	La aplicación tiene que ser capaz de Mostrar en tiempo real los detalles de todos los viajes activos.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 20: REQUISITOS DE CAPACIDAD (17)

Identificador	RUC- 18: Mostrar la lista de recogidas
Descripción	La aplicación tiene que ser capaz de Mostrar en tiempo real los detalles de la lista de recogidas que tiene cada conductor.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 21: REQUISITOS DE CAPACIDAD (18)

Identificador	RUC- 19: Mostrar la lista de entregas
Descripción	La aplicación tiene que ser capaz de Mostrar en tiempo real los detalles de la lista de recogidas que tiene cada conductor.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 22: REQUISITOS DE CAPACIDAD (19)

6.2.2 Requisitos de restricción

A continuación, se encuentran especificados los requisitos de usuario de restricción:

Identificador	RES-1: Lanzar la aplicación
Descripción	La aplicación se lanzará mediante la ejecución de un fichero .apk que se genera al compilar la aplicación.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 23: REQUISITOS DE CAPACIDAD (1)

Identificador	RES-2: Permisos
Descripción	Al instalar la aplicación el usuario debe otorgar el permiso de internet a la aplicación para que esta pueda funcionar de forma correcta en el teléfono.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 24: REQUISITOS DE CAPACIDAD (2)

Identificador	RES- 3: Acceso a la aplicación
Descripción	Para acceder a la aplicación es imprescindible que el usuario sea registrado en el sistema.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 25: REQUISITOS DE CAPACIDAD (3)

Identificador	RES-4: Layouts xml
Descripción	Los layouts de las pantallas se definen en archivos xml.
Necesidad	4
Prioridad	4
Estabilidad	Estable

TABLA 26: REQUISITOS DE CAPACIDAD (4)

6.3 Casos de uso

En esta sección se explicará más en detalle las funcionalidades que ofrece la herramienta para generar en el lector una visión clara de las posibilidades que ofrece este sistema y de su funcionamiento.

Los usuarios de la aplicación se pueden clasificar en tres tipos: Transportista, Recepcionista y el manager o jefe que es también un transportista, pero con más privilegios. Estos son los tipos que están definidos en la base de datos, de forma que sólo determinados tipos de usuarios pueden acceder a determinados módulos de la herramienta, según corresponda.

- **Identificador:** muestra el tipo de atributo CU (Caso de Uso) más un número secuencial que representará de forma unívoca el caso.
- **Descripción:** describe los pasos realizados por el usuario para la situación planteada.
- **Pre-condiciones:** condiciones que deben darse para la realización del caso de uso.
- **Post-condiciones:** condiciones que son resultado de la ejecución del caso de uso.

6.3.1 Operaciones que pueden realizar todos los usuarios

Registrar usuarios

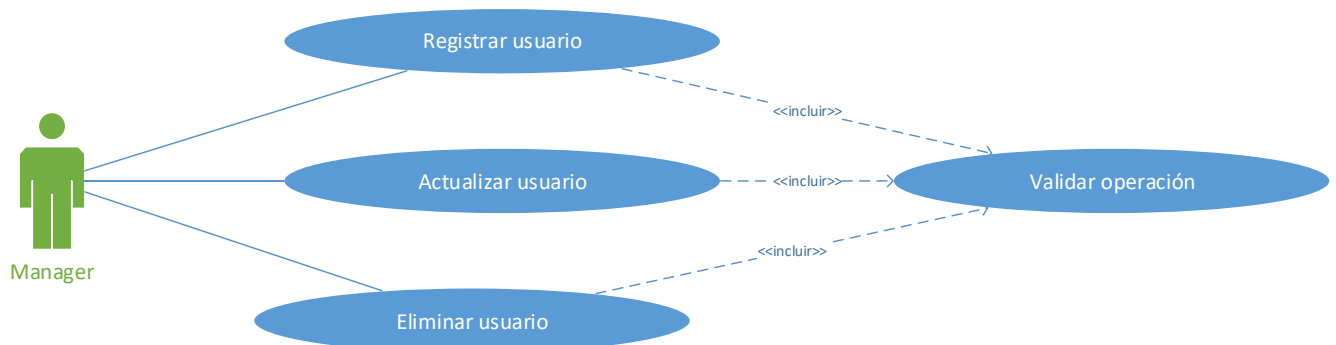


FIGURA 15: REGISTRAR USUARIOS

Identificador	CU-01
Nombre	Registrar usuario
Actores	El Manager
Descripción	<ul style="list-style-type: none"> • El manager inicia la aplicación • El manager rellena el formulario de registro • El manager pulsa el botón crear usuario
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet • El usuario tiene que ser manager
Post-condiciones	<ul style="list-style-type: none"> • Usuario registrado

TABLA 27: CASO DE USO (1)

Identificador	CU-02
Nombre	Actualizar usuario
Actores	El Manager
Descripción	<ul style="list-style-type: none"> • El manager inicia una sesión • El manager selecciona el usuario la lista • El manager rellena el formulario • El manager pulsa el botón actualizar usuario
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet • El usuario tiene que ser manager
Post-condiciones	Datos del usuario actualizados

TABLA 28: CASO DE USO (2)

Identificador	CU-03
Nombre	Eliminar usuario
Actores	El Manager
Descripción	<ul style="list-style-type: none"> • El manager inicia una sesión • El manager selecciona el usuario de la lista • El manager pulsa el botón eliminar
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet • El usuario tiene que ser manager
Post-condiciones	Usuario eliminado

TABLA 29: CASO DE USO (3)

INICIAR SESIÓN

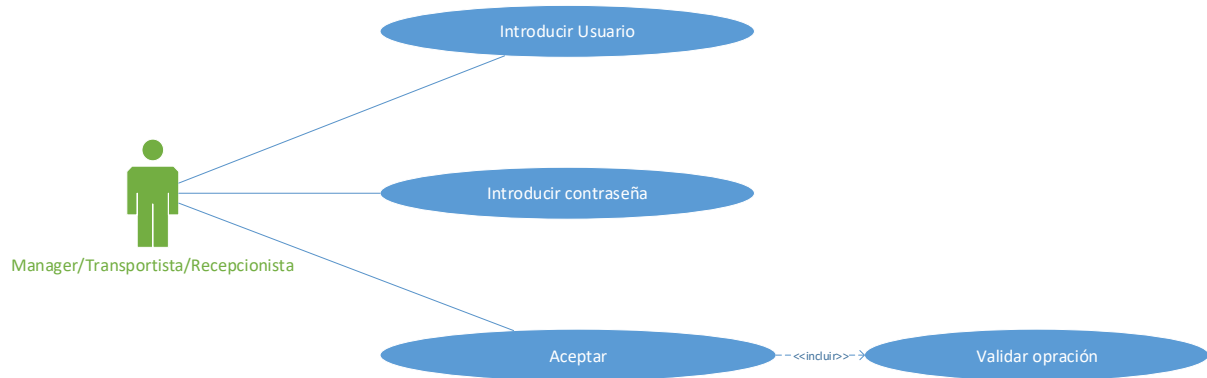


FIGURA 16: INICIAR SESIÓN

Identificador	CU-04
Nombre	Iniciar sesión
Actores	<ul style="list-style-type: none">• Manager• Transportista• Recepcionista
Descripción	<ul style="list-style-type: none">• El usuario Introduce usuario• El usuario introduce la contraseña• El usuario pulsa el botón aceptar
Pre-condiciones	<ul style="list-style-type: none">• Tener acceso a internet• El usuario tiene que estar registrado
Post-condiciones	Sesión iniciada

TABLA 30: CASO DE USO (4)

Gestión de clientes

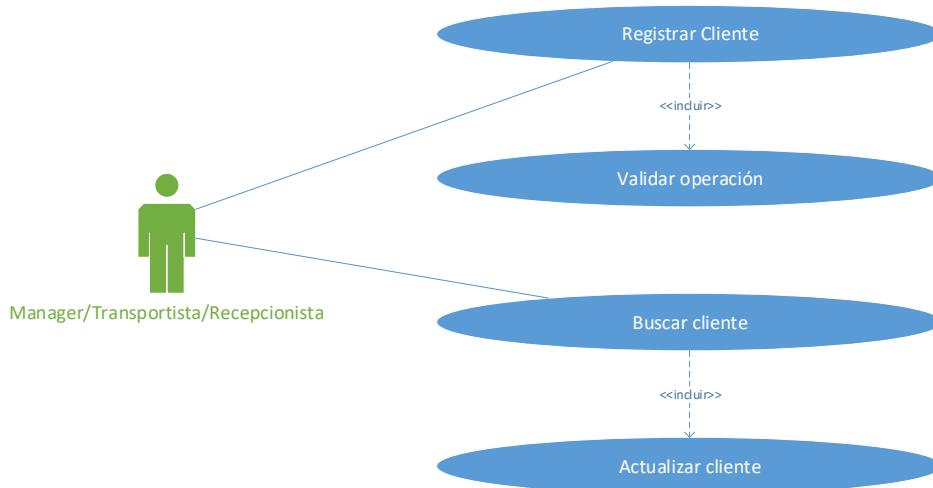


FIGURA 17: GESTIÓN DE CLIENTES

Identificador	CU-05
Nombre	Registrar cliente
Actores	<ul style="list-style-type: none"> • Manager • Transportista • Recepcionista
Descripción	<ul style="list-style-type: none"> • El usuario Introduce los datos del cliente en el formulario • El usuario pulsa el botón añadir cliente
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet • El cliente tiene que ser nuevo
Post-condiciones	El cliente está registrado

TABLA 31: CASO DE USO (5)

Identificador	CU-06
Nombre	Buscar cliente
Actores	<ul style="list-style-type: none"> • Manager • Transportista • Recepcionista
Descripción	<ul style="list-style-type: none"> • Filtrar por el nombre o DNI del cliente • Pulsar el botón buscar
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet • El cliente tiene que estar registrado • Los datos de búsqueda tienen que ser correctos
Post-condiciones	Se muestran los datos del cliente.

TABLA 32: CASO DE USO (6)

Identificador	CU-07
Nombre	Actualizar Cliente
Actores	<ul style="list-style-type: none"> • Manager • Transportista • Recepcionista
Descripción	<ul style="list-style-type: none"> • Seleccionar cliente • Actualizar los datos del cliente en el formulario • Pulsar el botón aceptar
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet • El cliente tiene que estar registrado
Post-condiciones	Los datos del cliente están actualizados.

TABLA 33: CASO DE USO (7)

Gestión de Viajes

Crear Viaje y añadir paquetes

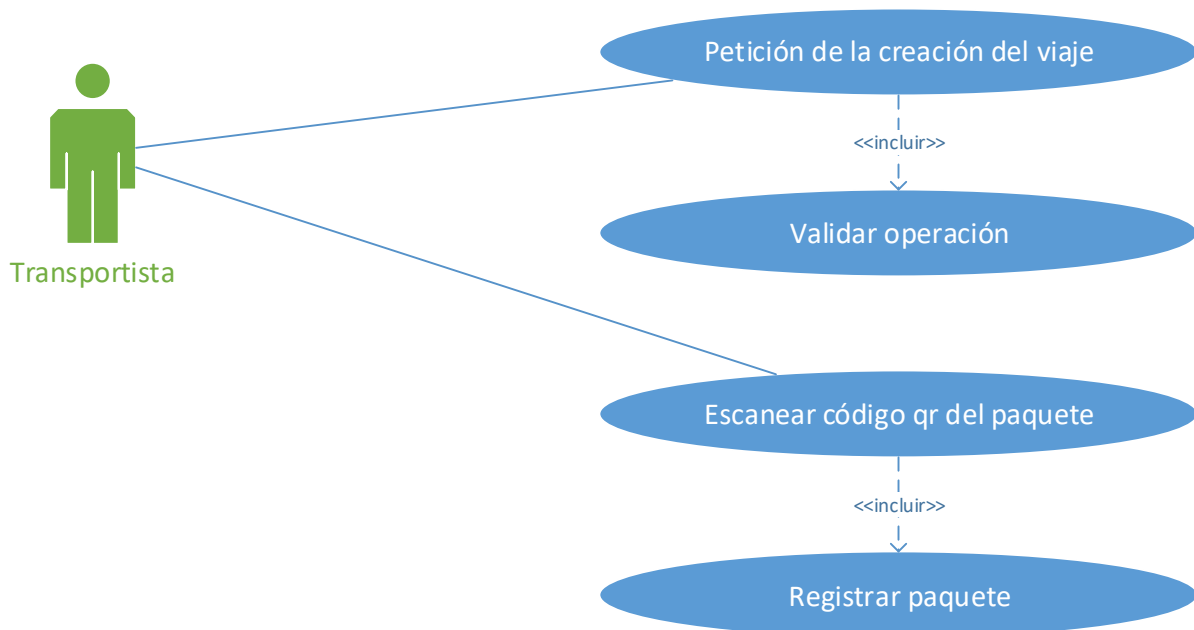


FIGURA 18: GESTIÓN DE VIAJES

Identificador	CU-08
Nombre	Petición de la creación de viaje
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El usuario introduce la ciudad de origen • El usuario introduce la ciudad de destino • El usuario pulsa el botón crear viaje
Pre-condiciones	<ul style="list-style-type: none"> • El usuario no tiene asignado un viaje • Tener acceso a internet
Post-condiciones	Viaje creado y asignado al usuario

TABLA 34: CASO DE USO (8)

Identificador	CU-09
Nombre	Escanear el código Qr de los paquetes
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El transportista pulsa el botón Escanear El código Qr Con el lector de código QR
Pre-condiciones	<ul style="list-style-type: none"> • El paquete tiene que estar registrado • El transportista tiene que tener un viaje asignado • Tener acceso a internet
Post-condiciones	<ul style="list-style-type: none"> • El paquete esta escaneado • El paquete esta asignado al viaje

TABLA 35: CASO DE USO (9)

Cerrar viaje

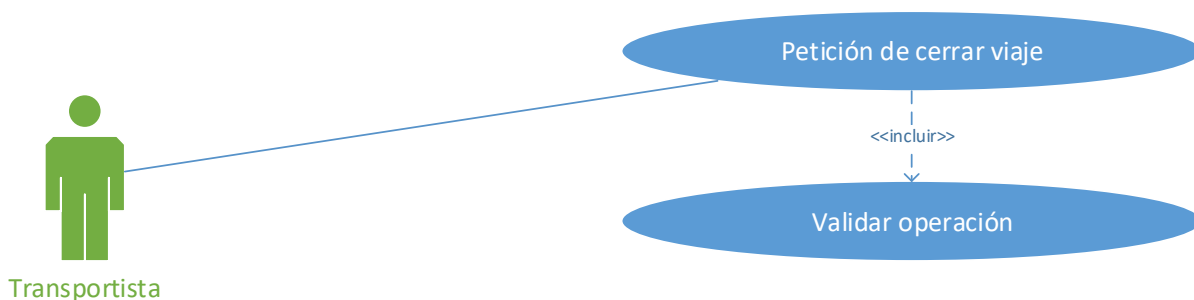


FIGURA 19: CERRAR VIAJE

Identificador	CU-10
Nombre	Petición de cerrar viaje
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El usuario selecciona la actividad cerrar viaje • El usuario pulsa el botón cerrar viaje
Pre-condiciones	<ul style="list-style-type: none"> • Tener un viaje asignado • Tener acceso a internet
Post-condiciones	Viaje cerrado

TABLA 36: CASO DE USO (10)

Modificar transportista de un viaje por otro

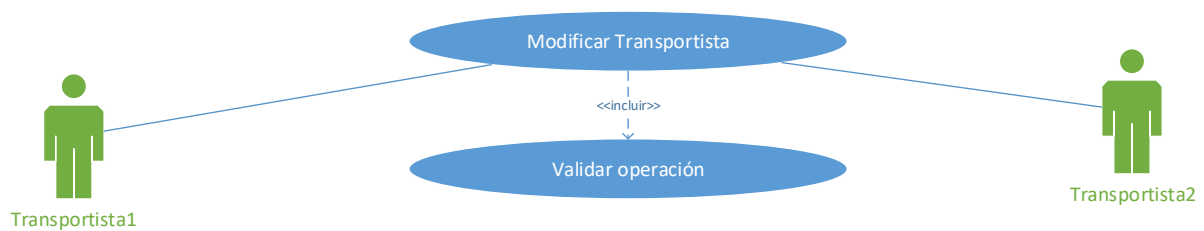


FIGURA 20: : MODIFICAR TRANSPORTISTA

Identificador	CU-11
Nombre	Modificar Transportista
Actores	<ul style="list-style-type: none"> • transportistas
Descripción	<ul style="list-style-type: none"> • El transportista original introduce su correo • El transportista original introduce su contraseña • El transportista nuevo introduce su correo • El transportista nuevo introduce su contraseña • El transportista original o nuevo pulsan el botón modificar transportista
Pre-condiciones	<ul style="list-style-type: none"> • El transportista nuevo no tiene un viaje asignado • El transportista original tiene un viaje asignado • Tener acceso a internet
Post-condiciones	<ul style="list-style-type: none"> • El viaje se asigna al nuevo transportista • El transportista original se libera del viaje

TABLA 37: CASO DE USO (11)

Añadir costes al viaje

Identificador	CU-12
Nombre	Añadir coste al viaje
Actores	<ul style="list-style-type: none">• Transportista
Descripción	<ul style="list-style-type: none">• El transportista introduce la cantidad del coste• El transportista introduce la descripción del coste• El transportista selecciona el tipo de la unidad monetaria
Pre-condiciones	<ul style="list-style-type: none">• El transportista tiene asignado un viaje• Tener acceso a internet
Post-condiciones	Coste añadido al coste total del viaje

TABLA 38: CASO DE USO (12)

Registrar un paquete y generar el código QR correspondiente

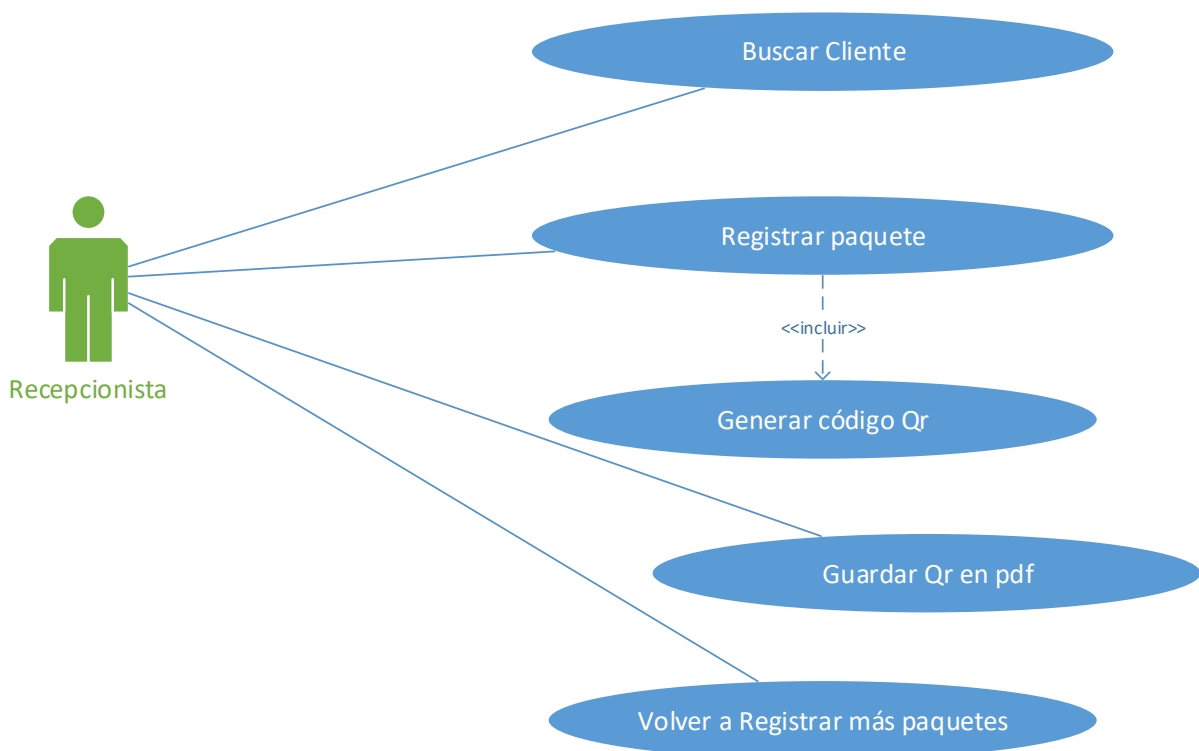


FIGURA 21: REGISTRAR UN PAQUETE

Identificador	CU-13
Nombre	Buscar cliente
Actores	<ul style="list-style-type: none"> • Recepcionista
Descripción	<ul style="list-style-type: none"> • Filtrar por el nombre o DNI del cliente • Pulsar el botón buscar
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet • El cliente tiene que estar registrado • Los datos de búsqueda tienen que ser correctos
Post-condiciones	Se muestran los datos del cliente.

TABLA 39: CASO DE USO (13):

Identificador	CU-14
Nombre	Registrar paquete
Actores	<ul style="list-style-type: none"> • Recepcionista
Descripción	<ul style="list-style-type: none"> • El recepcionista busca el cliente • El recepcionista selecciona el cliente de la lista • El recepcionista rellena el formulario de registro • El recepcionista pulsa el botón registrar mercancía
Pre-condiciones	<ul style="list-style-type: none"> • El cliente tiene que estar registrado • Tener acceso a internet
Post-condiciones	<ul style="list-style-type: none"> • El paquete registrado • El código QR esta generado para el paquete

TABLA 40: CASO DE USO (14)

Identificador	CU-15
Nombre	Guardar El código QR en formato pdf para poder imprimirlo
Actores	<ul style="list-style-type: none"> • Recepcionista
Descripción	<ul style="list-style-type: none"> • El recepcionista pulsa el botón guardar en pdf
Pre-condiciones	<ul style="list-style-type: none"> • El código Qr tiene que ser bien generado • Tener espacio suficiente de memoria en el teléfono móvil • Tener acceso a internet
Post-condiciones	<ul style="list-style-type: none"> • El Código Qr guardado en formato pdf

TABLA 41: CASO DE USO (15)

AÑADIR UN PAQUETE REGISTRADO AL VIAJE

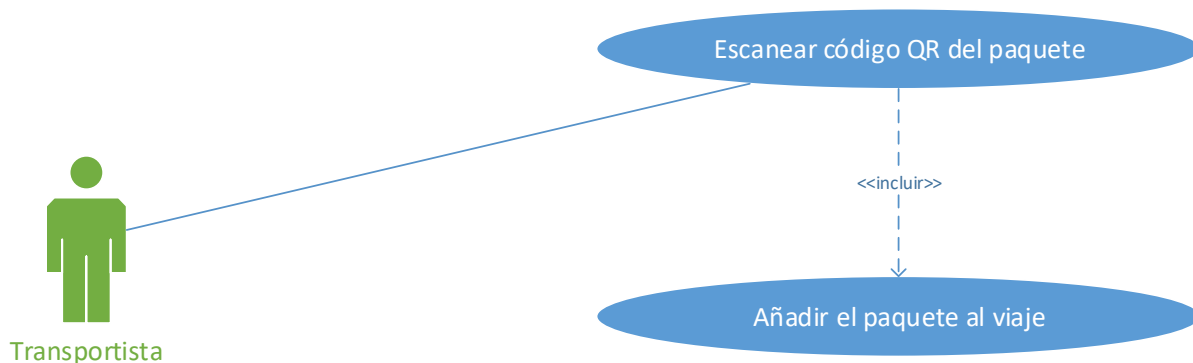


FIGURA 22:AÑADIR UN PAQUETE REGISTRADO AL VIAJE

Identificador	CU-16
Nombre	Escanear del código QR del paquete
Actores	<ul style="list-style-type: none">• Transportista
Descripción	<ul style="list-style-type: none">• Escanear el código QR del paquete• Asignar el paquete al viaje
Pre-condiciones	<ul style="list-style-type: none">• El paquete tiene que estar registrado• El viaje tiene que estar creado• Tener acceso a internet
Post-condiciones	Paquete asignado al viaje

TABLA 42: CASO DE USO (16)

Añadir un paquete no registrado al viaje

En este caso se registra primero el paquete generando su código QR, luego se escanea y se añade al viaje. Al final es la suma de los dos últimos casos de uso.

Pasar paquete a otro viaje

En esta operación pasamos un paquete de un viaje concreto a otro. Por ejemplo, se nuestro punto de origen es el punto A y nuestro destino es el punto C y vemos que hay dos viajes, el primero va del punto A al punto B y el otro viaje del punto B al C, aprovechamos los dos viajes para dejar el paquete en el destino.

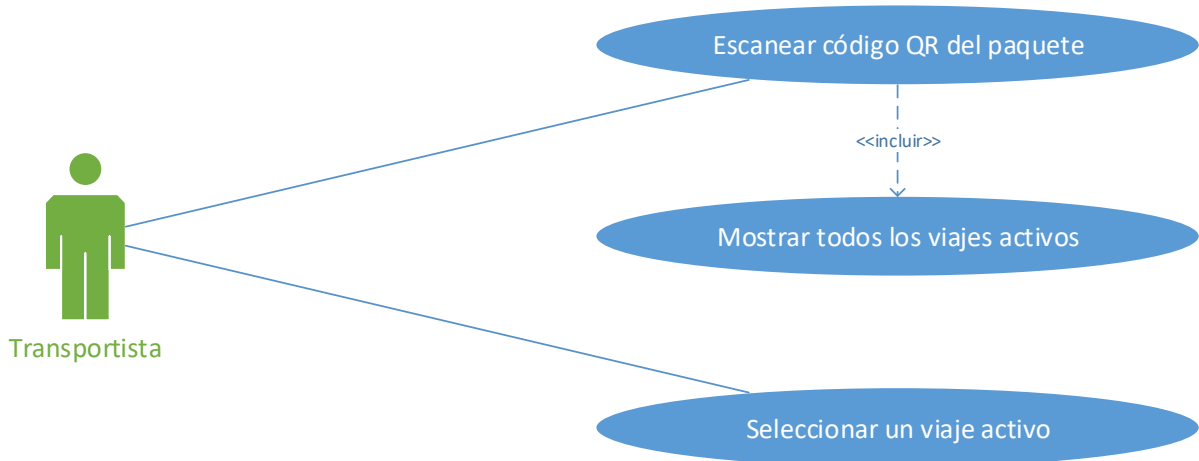


FIGURA 23: PASAR PAQUETE A OTRO VIAJE

Identificador	CU-17
Nombre	Pasar paquete a otro viaje
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El usuario escanea el código QR del paquete • El usuario busca el viaje • El usuario selecciona el viaje indicado • Tener acceso a internet
Pre-condiciones	<ul style="list-style-type: none"> • El paquete tiene que estar registrado en el viaje original
Post-condiciones	El paquete esta pasado al otro viaje

TABLA 43: CASO DE USO (17)

Entregar paquete

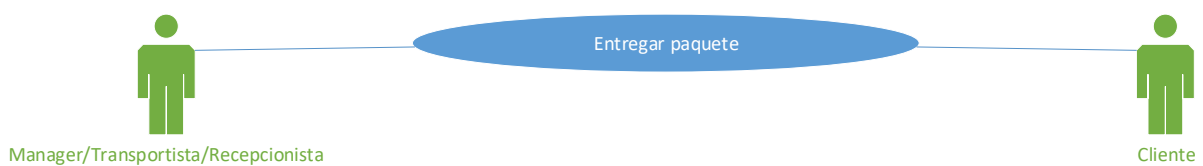


FIGURA 24: ENTREGAR PAQUETE

Identificador	CU-18
Nombre	Entregar paquete
Actores	<ul style="list-style-type: none"> • Transportista • Cliente
Descripción	<ul style="list-style-type: none"> • El transportista escanea el paquete • El transportista filtra por el nombre o el DNI del cliente • El transportista pulsa el botón entregar paquete
Pre-condiciones	<ul style="list-style-type: none"> • Llegar a la dirección del cliente • Tener acceso a internet
Post-condiciones	Paquete entregado.

TABLA 44: CASO DE USO (18)

GESTIÓN DE INCIDENCIAS

Crear una nueva incidencia

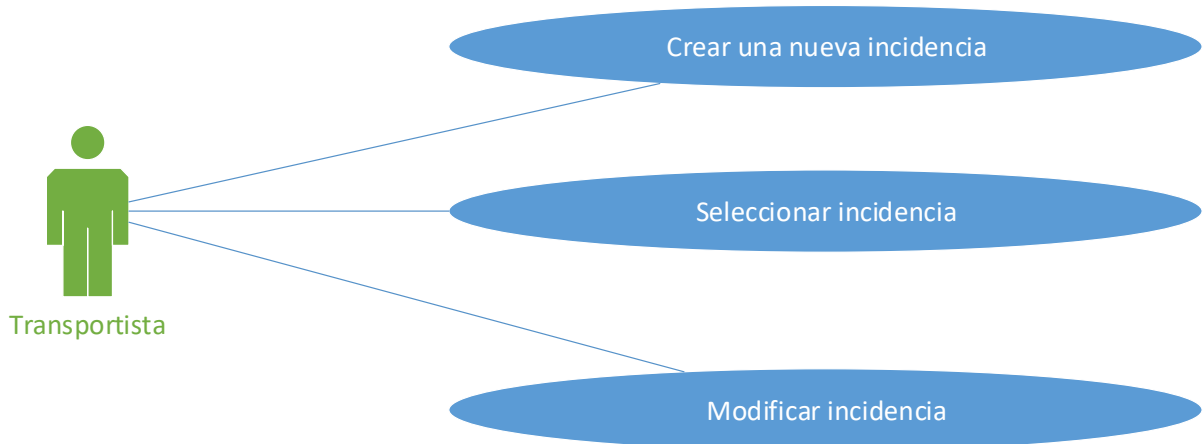


FIGURA 25: CREAR NUEVA INCIDENCIA

Identificador	CU-19
Nombre	Crear una nueva incidencia
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El transportista rellena el formulario • El transportista pulsa el botón nueva incidencia
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet
Post-condiciones	Incidencia creada

TABLA 45:: CASO DE USO (19)

Modificar datos de la incidencia

Identificador	CU-20
Nombre	Modificar incidencia
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El transportista selecciona la incidencia de la lista • El transportista modifica los datos de la incidencia
Pre-condiciones	<ul style="list-style-type: none"> • La incidencia tiene que estar creada primero • Tener acceso a internet
Post-condiciones	<ul style="list-style-type: none"> • Incidencia modificada

TABLA 46: CASO DE USO (20)

Modificar el transportista de un viaje

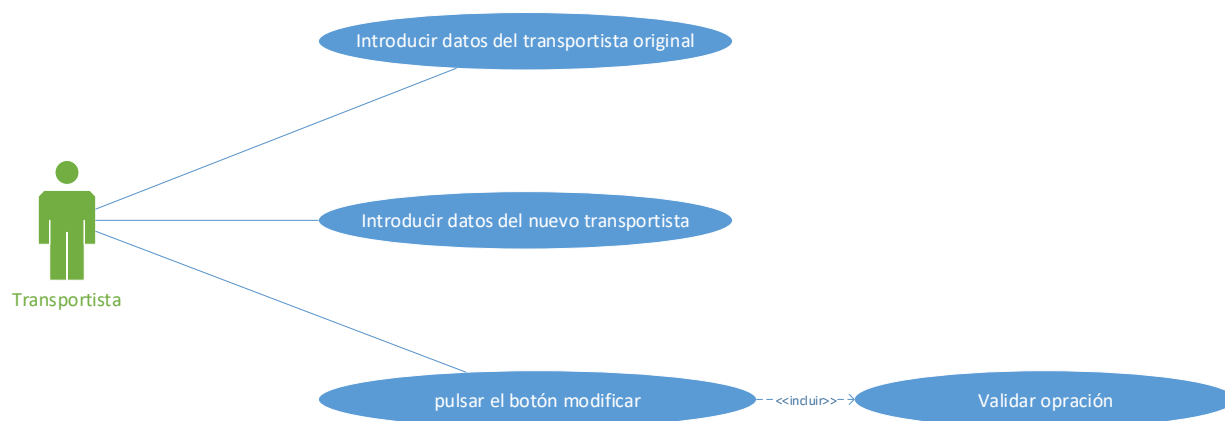


FIGURA 26: MODIFICAR TRANSPORTISTA

Identificador	CU-21
Nombre	Modificar transportista
Actores	<ul style="list-style-type: none"> • transportista
Descripción	<ul style="list-style-type: none"> • El transportista introduce sus datos • El transportista introduce los datos del nuevo transportista • El transportista pulsa el botón modificar
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet
Post-condiciones	<ul style="list-style-type: none"> • Transportista modificado

TABLA 47: MODIFICAR TRANSPORTISTA (21)

Registrar futuras recogidas

En esta operación el recepcionista registra todas las recogidas futuras recibiendo llamadas de los clientes, de esta manera obtenemos una lista general de recogidas que puede ser vista desde todos los viajes, entonces cuando algún transportista quiere añadir un paquete a su viaje, lo que hace es seleccionar los paquetes de esta lista, que aparecerán automáticamente en una lista de recogidas que ve el solo y dejan de verse de la lista general.

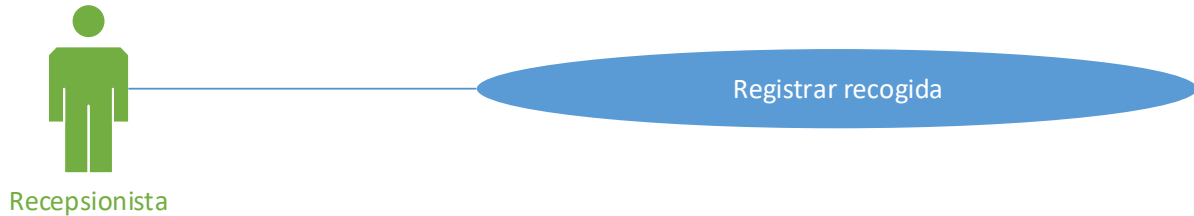


FIGURA 27: REGISTRAR FUTURAS RECOGIDAS

Identificador	CU-22
Nombre	Registrar recogida
Actores	<ul style="list-style-type: none">• Recepcionista
Descripción	<ul style="list-style-type: none">• El recepcionista busca el cliente• El recepcionista selecciona el cliente• El recepcionista rellena el formulario de la recogida• El recepcionista pulsa el botón registrar recogida
Pre-condiciones	<ul style="list-style-type: none">• El cliente tiene que estar registrado• El usuario tiene que tener acceso a internet
Post-condiciones	<ul style="list-style-type: none">• Recogida registrada

TABLA 48: MODIFICAR TRANSPORTISTA (22)

Mostrar todas las recogidas en una lista general

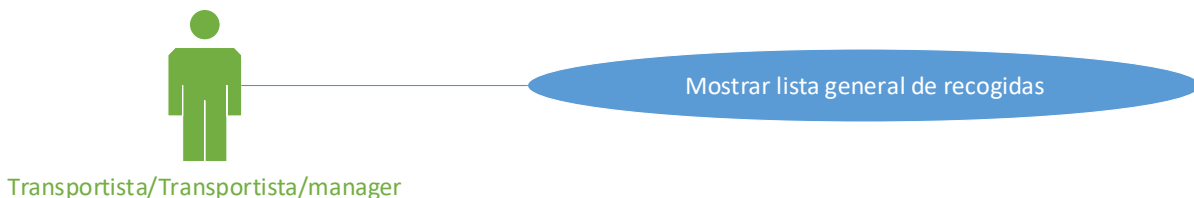


FIGURA 28: LISTA GENERAL DE RECOGIDAS

Identificador	CU-23
Nombre	<ul style="list-style-type: none"> Mostrar lista general de recogidas
Actores	<ul style="list-style-type: none"> Manager Transportista Recepcionista
Descripción	<ul style="list-style-type: none"> Mostrar lista general de recogidas
Pre-condiciones	<ul style="list-style-type: none"> Tener acceso a internet
Post-condiciones	Se muestran las recogidas según el filtro

TABLA 49: MODIFICAR TRANSPORTISTA (23)

Mostrar todos los viajes activos

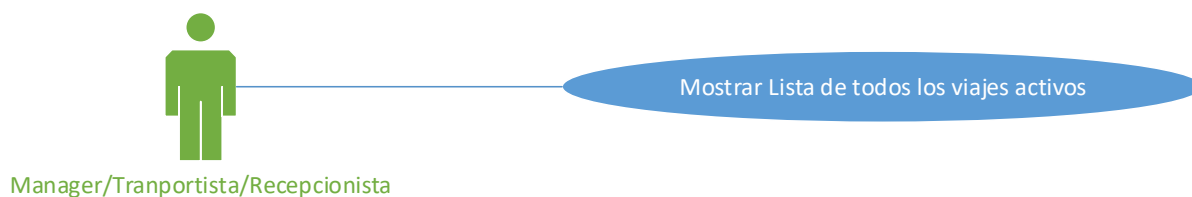


FIGURA 29: LISTA DE VIAJES ACTIVOS

Identificador	CU-24
Nombre	Lista de viajes activos
Actores	<ul style="list-style-type: none"> Recepcionista Transportista Manager
Descripción	<ul style="list-style-type: none"> El usuario pulsa el botón viajes activos El usuario muestra la lista de viajes activos
Pre-condiciones	<ul style="list-style-type: none"> Tener acceso a internet
Post-condiciones	<ul style="list-style-type: none"> Se muestran la lista de viajes activos

TABLA 50: MODIFICAR TRANSPORTISTA (24)

Descubrir paquete

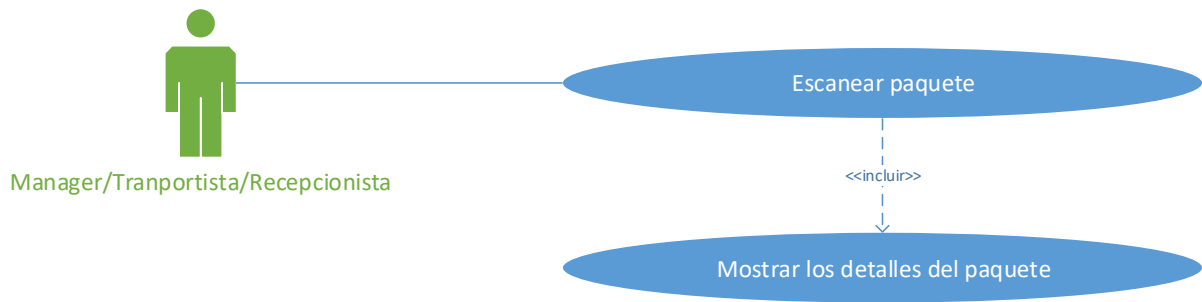


FIGURA 30: DESCUBRIR PAQUETE

Identificador	CU-25
Nombre	Descubrir paquete
Actores	<ul style="list-style-type: none"> • Manager • Transportista • recepcionista
Descripción	<ul style="list-style-type: none"> • El usuario escanea el código QR del paquete • En la pantalla se muestran los datos del paquete
Pre-condiciones	<ul style="list-style-type: none"> • El paquete tiene que estar registrado • Tener acceso a internet
Post-condiciones	<ul style="list-style-type: none"> • Datos del paquete mostrados

TABLA 51: MODIFICAR TRANSPORTISTA (25)

Lista de incidencias de un viaje

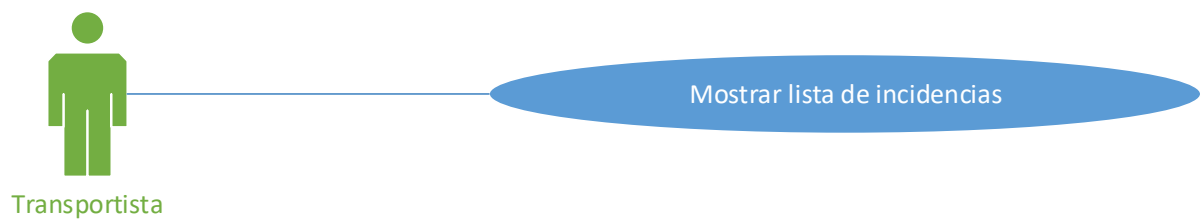


FIGURA 31: DESCUBRIR PAQUETE

Identificador	CU-26
Nombre	Mostrar lista de incidencias
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El usuario pulsa el botón listo de incidencias • Se muestra la lista de incidencias

Pre-condiciones	<ul style="list-style-type: none"> • Tener incidencias registradas • Tener acceso a internet
Post-condiciones	Se muestran las listas de incidencias

TABLA 52: MODIFICAR TRANSPORTISTA (26)

Lista de mercancías de un viaje

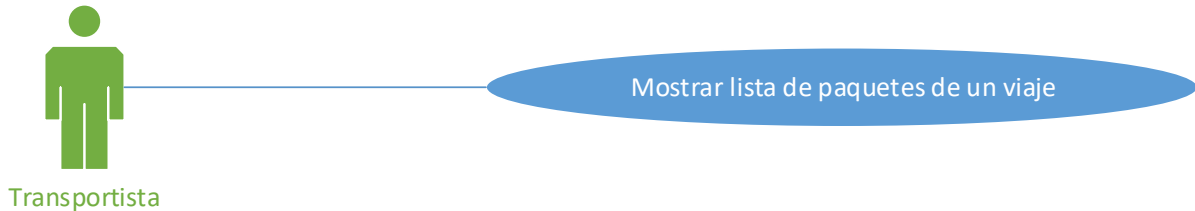


FIGURA 32: LISTA DE MERCANCÍAS DE UN VIAJE

Identificador	CU-27
Nombre	Mostrar lista de paquetes de un viaje
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El usuario pulsa el botón mercancías • Mostrar lista de paquetes de un viaje
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet • Tener paquetes registrados para el viaje
Post-condiciones	Mostrados la lista de paquetes del viaje

TABLA 53: MOSTRAR LISTA DE PAQUETES DE UN VIAJE (27)

Lista de entregas

En la lista de entregas se muestran dos tipos paquetes entregados con icono verde y otras pendientes de entregas con icono de color rojo.

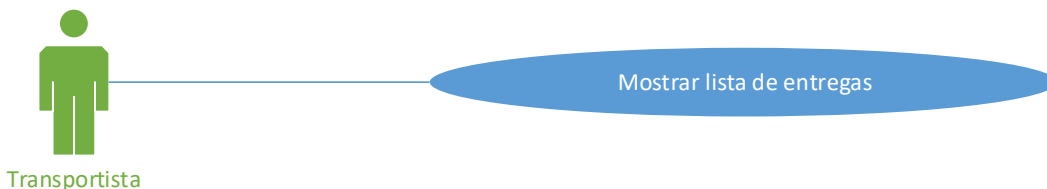


FIGURA 33: LISTA DE ENTREGAS

Identificador	CU-28
Nombre	Lista de entregas
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El usuario pulsa el botón lista de entregas • Se muestran la lista de entregas
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet
Post-condiciones	Se muestran la lista de entregas

TABLA 54: MOSTRAR LISTA DE PAQUETES DE UN VIAJE (28)

Lista de recogidas del transportista



FIGURA 34: LISTA DE RECOGIDAS DEL TRANSPORTISTA

Identificador	CU-29
Nombre	Lista de recogidas del transportista
Actores	<ul style="list-style-type: none"> • Transportista
Descripción	<ul style="list-style-type: none"> • El transportista pulsa el botón mostrar recogidas como consecuencia se muestran las recogidas del transportista
Pre-condiciones	<ul style="list-style-type: none"> • Tener acceso a internet
Post-condiciones	Se muestran la lista de recogidas del conductor.

TABLA 55: MOSTRAR LISTA DE RECOGIDAS DEL TRANSPORTISTA (29)

6.4 Diagrama de clases

En este apartado se definen las clases a utilizar y cuáles son las relaciones entre ellas. Según las buenas prácticas para programar una aplicación móvil con Android, se necesita una clase por cada vista o pantalla. Además de estas clases, se tienen otras clases auxiliares necesarias para la correcta gestión de la aplicación. El diagrama de clases representado en la siguiente figura muestra todas las que se han definido en este trabajo.

Todas las clases que extienden de Activity han de ir definidas en un archivo .xml llamado AndroidManifest. Una Activity es una entidad sencilla que se usa para llevar a cabo acciones. Una aplicación puede tener como es el caso, muchas actividades diferentes, pero el usuario sólo interactúa con ellas de una en una.



FIGURA 35: DIAGRAMA DE CLASES

6.5 Diseño

Con los datos obtenidos durante la fase de análisis, se realiza un diseño de los componentes, de forma que todos los aspectos queden claros y definidos para la fase de implementación. Con toda esta información, durante la fase de implementación no será necesario replantear ningún punto. El diseño se ha de adaptar a las necesidades de la aplicación, y las clases se han de definir de manera exhaustiva, de manera que las relaciones entre ellas queden claras.

6.5.1 Arquitectura del software

A raíz de lo descrito puede deducirse que la arquitectura más conveniente para el sistema es la cliente-servidor.

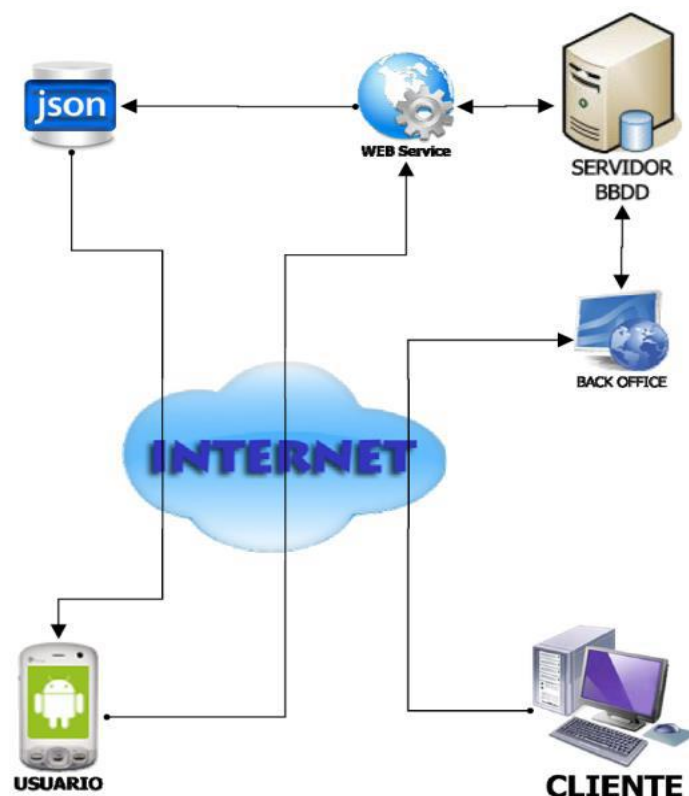


FIGURA 36: ARQUITECTURA DEL SOFWATRE

Este tipo de arquitectura es típica en sistema distribuidos (como pueden serlo las instalaciones domóticas con sus sensores, actuadores y centralitas), en los que los datos y el procesamiento se distribuye en módulos independientes pero interconectados. Los componentes de dicha arquitectura son:

- Servidores que ofrecen servicios a otros sistemas. Estos servidores, por lo general, no conocen a los sistemas que solicitan sus servicios. Un ejemplo típico de este tipo de sistemas son los servidores web, que sirven una gran cantidad de datos a un conjunto

de clientes muy variado (pueden ser navegadores de escritorio, clientes FTP, bots de buscadores, etcétera).

- Clientes que utilizan los servicios ofrecidos por los servidores. Estas solicitudes e invocaciones pueden llegar a ejecutarse masivamente y de forma concurrente. Los clientes sí tienen la responsabilidad de saber qué servidores ofrecen qué contenidos y cómo tienen que invocar a sus servicios. Por ejemplo, cuando usamos un navegador web tenemos que saber qué página queremos visitar.
- Una red de interconexión que permite a los clientes acceder a los servicios.

Debido a la naturaleza cliente-servidor del sistema a desarrollar, es importante realizar un diseño que consiga una alta cohesión y un bajo acoplamiento. Estas dos cualidades nos permiten reducir la complejidad de los subsistemas y, por tanto, del sistema en general. Esta reducción de complejidad es beneficiosa para el mantenimiento del sistema, su comprensión y su rendimiento.

6.5.2 Diseño de la base de datos

Para proceder a la creación de la base de datos, primero se ha estudiado el universo de discurso para generar el diagrama entidad-relación del modelo: seleccionando los conjuntos entidad, describiendo las propiedades de cada conjunto entidad y seleccionando los conjuntos asociación.

En la figura se muestra el diagrama ER a partir del cual se construirán las sentencias SQL necesarias para la generación de la base de datos.

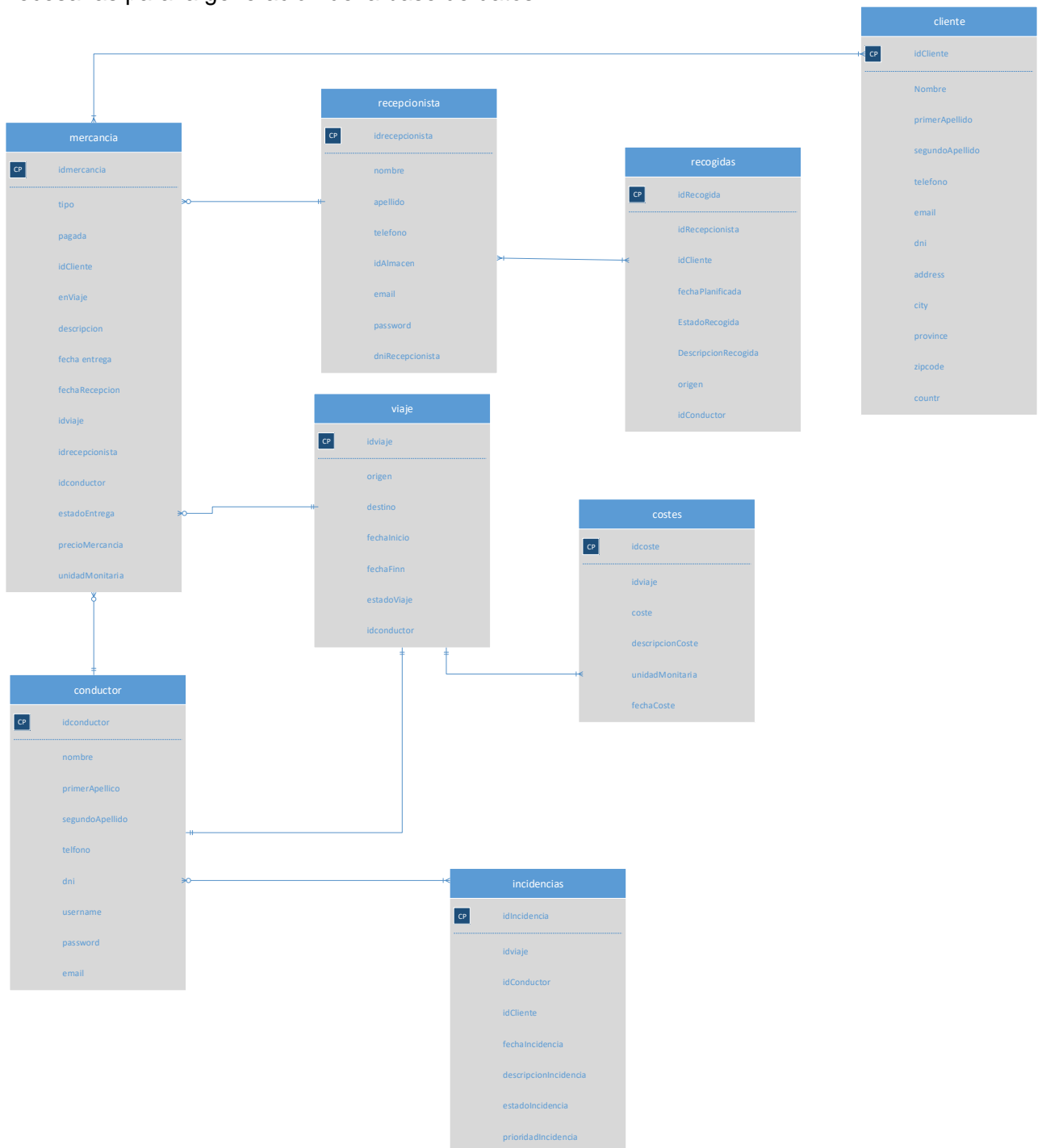


FIGURA 37: DISEÑO DE LA BASE DE DATOS

7. Conclusiones y trabajos futuros

En este capítulo se exponen las conclusiones obtenidas tras la realización del trabajo, haciendo un repaso global del mismo y comparando los resultados obtenidos con los objetivos marcados inicialmente.

7.1 Conclusiones

El estudio sobre las tecnologías a emplear, análisis y diseño de la herramienta, implementación en código y periodo de pruebas han sido las principales fases de este trabajo.

Tras la conclusión de todas estas etapas que han constituido el desarrollo del trabajo, es momento de hacer balances sobre los resultados obtenidos. El objetivo principal marcado en el inicio de este trabajo fue construir una herramienta que permitiera agilizar las actividades de gestión de expediciones de una empresa de mercancías. Dicho objetivo puede decirse que se ha cumplido satisfactoriamente. A continuación, se hace un repaso del estado de los componentes que forman la herramienta:

Cuando se inició este trabajo se plantearon una serie de objetivos genéricos ligados al prototipo que se quería desarrollar. Sin embargo, el propio proceso de desarrollo también tenía sus propios objetivos, siendo el primer objetivo fundamental es aprender más sobre la plataforma de Android.

La búsqueda de una aplicación que fuera suficientemente útil para ser usada, abordable por una persona en un plazo de tiempo razonable y variada como para aprender los aspectos típicos de esta plataforma no fue en absoluto trivial, pero fue un reto que me permitió no solo poner en práctica la mayoría de los conocimientos aprendidos durante mis estudios, sino también mejorar mi capacidad de autoaprendizaje.

La tecnología más innovadora o moderna del trabajo es la referente a la plataforma Android. Personalmente como desarrollador del trabajo he disfrutado aprendiendo y programando en esta nueva plataforma. Aunque en un principio el avance es lento, al final tras comprender la filosofía de la plataforma y todo lo que la rodea, se pone de manifiesto la potencia que tiene dicho sistema con mecanismos tan novedosos como los intents o la fácil integración de otros servicios. El balance referente al uso de esta plataforma es muy positivo. Las características que llevan a esta conclusión son las siguientes:

- Android es open source, lo cual significa que cualquier persona puede contribuir a mejorar el sistema.
- Tiene el respaldo de las empresas más importantes, tanto a nivel de fabricantes, como operadoras.
- El crecimiento de la plataforma es constante y se puede decir que está en pleno auge.
- Su aplicación "Google play" pone a disposición de los usuarios una amplia gama de aplicaciones de todo tipo.
- Google ha realizado un buen trabajo en cuanto a documentar la plataforma, centralizando toda la información útil para los desarrolladores en su página web

<https://developer.android.com>. A parte, actualmente la comunidad Android en internet ha crecido mucho, de forma que es posible encontrar blogs, foros y webs sobre desarrollo donde obtener información igualmente muy útil.

- El hecho de que el lenguaje usado para desarrollar sobre esta plataforma sea Java con archivos xml para los interfaces de usuario facilita mucho la programación, dado que ya se tiene este conocimiento de forma previa a iniciarse en Android.
- La construcción de los interfaces gráficos se ve facilitada gracias al entorno gráficos que se proporciona para ello, además de su sistema de archivos .xml que se usan para tal fin.
- Cabe destacar, por supuesto, lo fácil que es integrar una aplicación con los servicios que proporciona Google, lo cual dota de mayor potencia a las aplicaciones desarrolladas para este sistema.

La confirmación por parte del usuario que se impone para utilizar ciertas capacidades del teléfono por parte de aplicaciones de terceros proporciona un nivel de seguridad a la plataforma carente en otros sistemas.

7.2 Trabajos futuros

Como líneas de trabajo futuras, se propone ampliar y mejorar la funcionalidad de la herramienta con las siguientes sugerencias:

Debido a que el módulo del cliente móvil ha sido desarrollado sobre la plataforma Android, su usabilidad está limitada por tanto a los terminales con dicho sistema operativo. La implementación de un interfaz web móvil transformará la aplicación en multiplataforma, de forma que podrá ser utilizada desde terminales de otro tipo y con otro sistema operativo.

Una mejora que será muy útil para el transportista será poder visualizar las expediciones ordenadas secuencialmente de forma que el tiempo empleado en completar su hoja de ruta sea el menor posible. Esto deberá implementarse mediante algún tipo de algoritmo a nivel servidor, el cual enviará al dispositivo móvil las expediciones ya ordenadas. Esto tendrá además la ventaja añadida de un ahorro considerable de combustible por el vehículo utilizado, reduciendo así costes.

La integración de otros módulos y aplicaciones Android en la herramienta complementarían las funciones que ya ofrece y aumentaría la utilidad de la misma. Incluso podrían definirse varios perfiles móviles según el tipo de reparto que se vaya a realizar. Por ejemplo, mostrar aplicaciones integradas diferentes para un repartidor a pie y para un repartidor con vehículo motorizado.

Sería muy útil añadir la geolocalización de los medios de transporte en tiempo real, de esta manera se pueden coordinar bien las entregas y recogidas.

Sería muy útil también añadir una nueva pantalla que permite que el cliente firme después de recibir su paquete, de esta manera se puede obtener una prueba física de la entrega.

Me gustaría también añadir un módulo de estadística que permite visualizar graficas de los ingresos, costes, aumento del número de clientes, número de paquetes distribuidos, ver los clientes que más usan nuestros servicios, etc.

Incluir el perfil de usuario para el seguimiento de sus pedidos y para pedir recogidas.

Finalmente, se podrán realizar todo tipo de actualizaciones en la aplicación que representarán pequeñas mejoras de cada al usuario: nuevos diseños de interfaces gráficos más atractivos y configurables por el usuario, más opciones disponibles como, por ejemplo, mostrar en el mapa sólo las expediciones más cercanas a una determinada posición, adición de nuevos idiomas para seleccionar, etc.

8. Presupuesto

Se presentan a continuación los cálculos del coste. Para la realización de los mismos se ha tomado en cuenta los costes tanto de material como de mano de obra. Además, se añade también una sección sobre el coste por mantenimiento.

8.1. Coste de material

El material que se ha utilizado en el desarrollo de este trabajo ha sido:

- Ordenador portátil Lenovo G50-80 con un procesador Intel Core i7-5500U 2.4 GHz, memoria de 8 GB y 1TB de almacenamiento
- Dispositivo móvil Vernee Mix 2, con un procesador MTK6757CD Octa Core 2,5 GHz, memoria RAM de 4 GB
- Ordenador de sobremesa usado como servidor de un procesador de 3,25 GHz, 4 GB de memoria

En cuanto al software, se han utilizado los siguientes programas:

- Android studio
- Ubuntu Server
- Mysql
- Apache
- Librería Android Volley para enviar y recibir respuestas Json
- Windows 10

Todo el software mencionado es Gratuito (el Windows 10 fue proporcionado por la universidad)

CONCEPTO	PRECIO
Ordenador portátil	550
Ordenador sobremesa (de segunda mano)	100
Móvil	180
Total	830

TABLA 56: TABLA DE COSTES DE MATERIAL

El coste total de los materiales es de 830 euros.

8.2. Coste de mano de obra

Por la imposibilidad de haber tenido una dedicación a tiempo completo al trabajo, debido a motivos laborales y familiares, el desarrollo del mismo se ha extendido bastante en el tiempo.

Los gastos de personal se han calculado teniendo en cuenta los siguientes aspectos:

- Se han dedicado una media de 2 horas al día por el trabajo.

Teniendo en cuenta estas consideraciones y la planificación del apartado anterior, los días de dedicación real al trabajo para cada actividad se muestran en la siguiente tabla:

Actividad	Días reales
Fase Creativa	2
Estudio de las posibles plataformas	3
Estudio de la plataforma seleccionada	8
Documentación inicial	9
Tutorialización y demos	10
Diseño de la aplicación	8
Implementación	65
Documentación	65
Pruebas	12

TABLA 57: DIAS REALES DE TRABAJO

Horas totales = $(2+3+8+9+10+8+65+65+12) * 2 = 364$ horas

Consideramos que salario medio de un ingeniero es de 15 euros la hora

Total, días = 182 días

Horas diarias = 2 horas

Total = $182 * 2 * 15 = 5460$ euros

8.3 Coste total

Una vez calculados los costes de los conceptos anteriores, se puede determinar el presupuesto para este trabajo, que se muestra en la siguiente tabla:

Costes	Presupuesto (euros)
Costes de material	830
Costes de mano de obra	5460
Total	6290

TABLA 58: COSTES TOTALES

En total el presupuesto de este trabajo se estima en nueve mil doscientos euros

9. Resultados

En este capítulo, se muestran las diferentes interfaces que componen la aplicación Transpack. Una vez instalada la aplicación en el terminal, el usuario podrá ejecutarla desde el menú de aplicaciones de Android. En la siguiente pantalla se muestra el menú de aplicaciones con el icono de la aplicación Transpack.

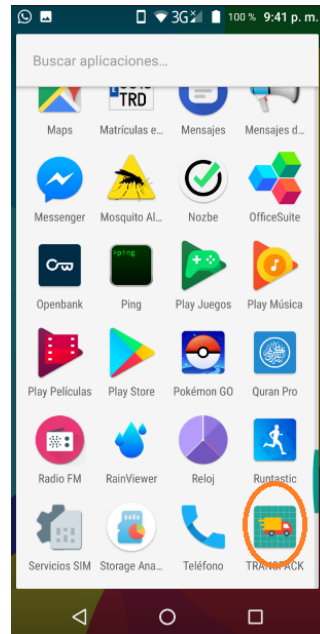


FIGURA 38: MEN U DE APLICACIONES DEL DISPOSITIVO.

Pantalla de bienvenida

La primera pantalla que aparece después de la instalación es la pantalla de bienvenida, sirve para mostrar el logo de la aplicación durante unos segundos.



TRANSPACK



Figura 39: pantalla de bienvenida

Pantalla de inicio de sesión

La pantalla de inicio de sesión nos permite acceder a la aplicación rellenando el correo y la contraseña previamente al pulsar el botón Aceptar, en el caso contrario se muestra un mensaje indicando que tenemos que rellenar los campos vacíos. En el caso de que los datos introducidos son incorrectos se muestra un mensaje que lo indica.



FIGURA 40: INICIO DE SESIÓN

Menú principal de la aplicación

Este primer menú es el responsable de gestionar todos los componentes de la aplicación.

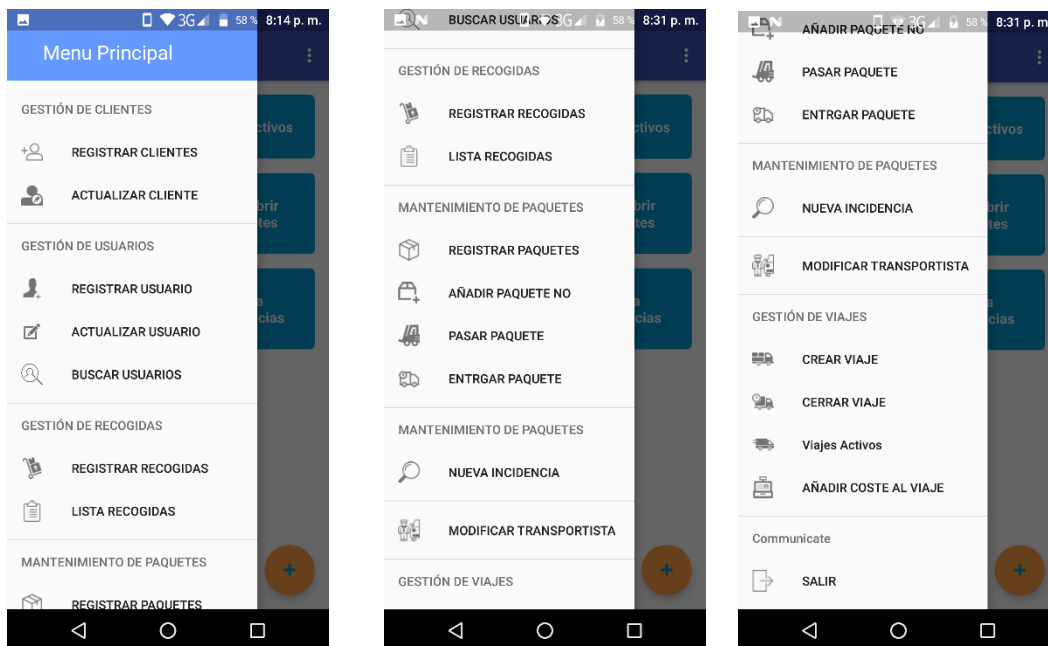


FIGURA 41: PRIMER MENÚ PRINCIPAL

El segundo menú es el responsable de mostrar los datos relacionados con la expedición que tiene asignada el Transportista.



FIGURA 42: SEGUNDO MENÚ PRINCIPAL

Registrar Cliente (puede ser realizada por cualquier usuario)

La siguiente pantalla sirve para registrar clientes, se introducen los datos de cliente y se pulsa el botón registrar. Esta pantalla contiene un control de los campos vacíos. Después de pulsar el botón saldrá un mensaje indicando que el cliente fue registrado con éxito en contra saldrá otro que indica lo contrario.

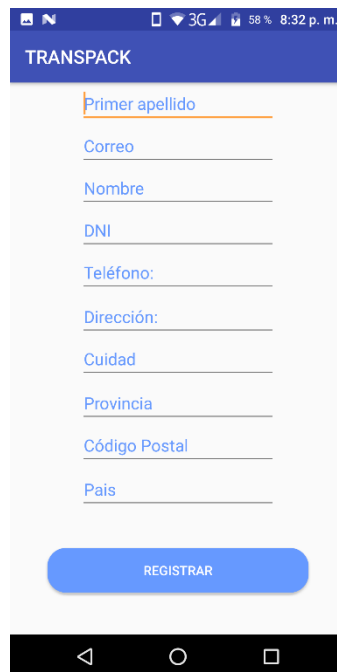


FIGURA 43:REGISTRO DE CLIENTES

Actualizar cliente

En las siguientes dos pantallas, primero buscamos y seleccionamos un cliente de la lista de clientes filtrando o no, y nos aparece otra pantalla que muestra todos los datos del cliente donde podemos actualizar los datos pulsando el botón Actualizar.

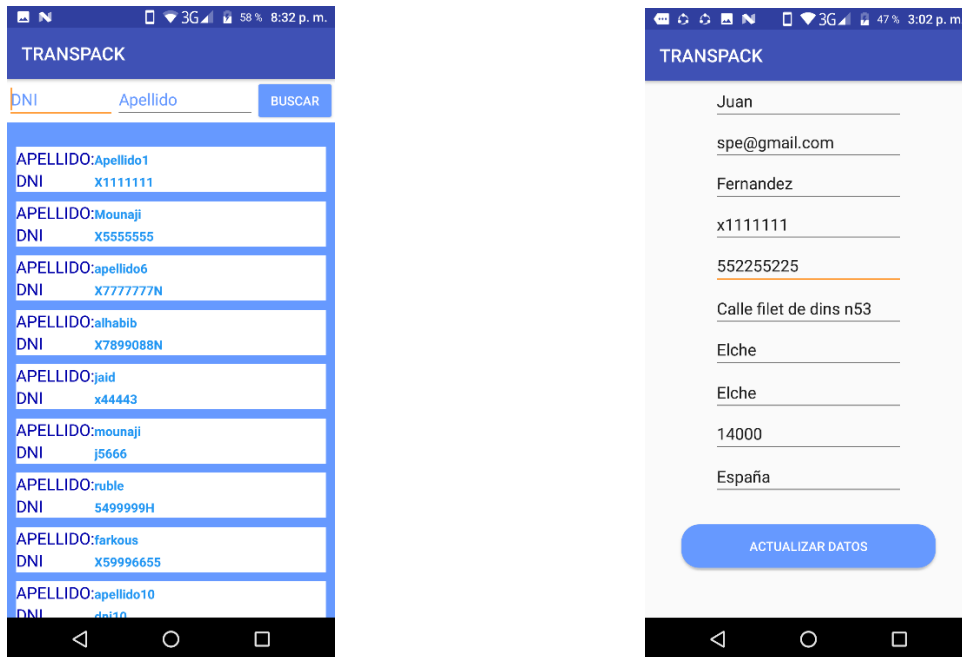


FIGURA 44: ACTUALIZACIÓN DE LOS DATOS DE UN CLIENTE

Registrar usuario (puede ser realizada solo por el manager)

En esta pantalla rellenamos los datos del usuario y pulsamos el botón registrar usuario. La pantalla contiene un control si algunos campos se quedan vacíos y muestra un mensaje indicando si la acción de registro se efectuó con éxito o no.

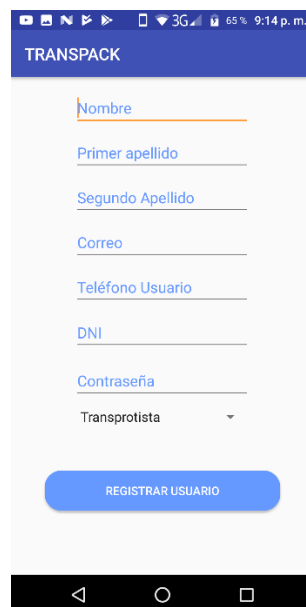


FIGURA 45: REGISTRO DE USUARIO

Actualizar usuario (solo puede ser realizado por el manager)

En la primera pantalla buscamos y seleccionamos el usuario y nos aparece la segunda pantalla donde aparecen los datos del usuario, actualizamos los datos y pulsamos el botón actualizar usuario.

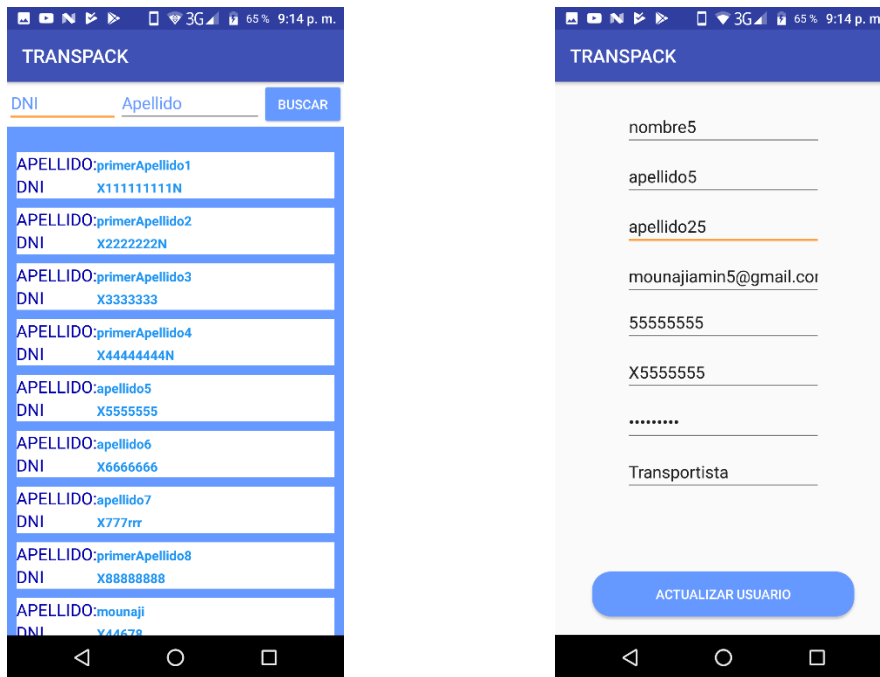


FIGURA 46: ACTUALIZAR USUARIO

Registrar recogidas (solo puede ser realizada por el recepcionista)

En la primera pantalla buscamos y seleccionamos el cliente, aparece la segunda pantalla donde metemos los datos de la recogida. Cuando pulsamos el botón seleccionar fecha no aparece el calendario donde podemos seleccionar día, mes y año.

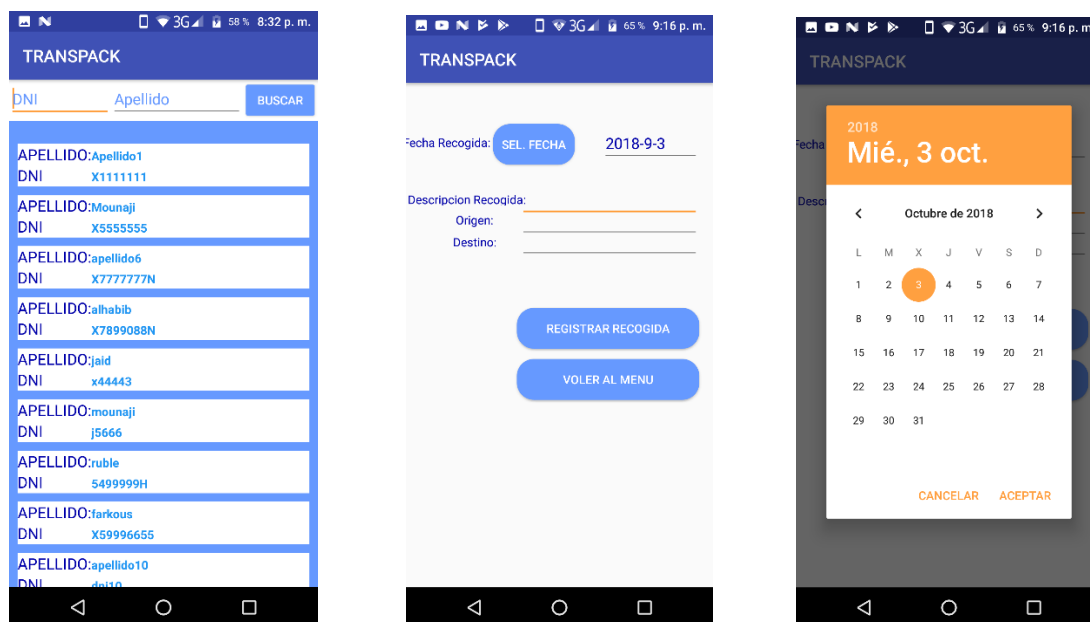


FIGURA 47: REGISTRO DE RECOGIDA DE PAQUETES

Lista general de recogidas

En Esta pantalla aparecen las recogidas de paquetes registradas. Esta lista esta vista por todos los transportistas donde pueden seleccionar recogidas que están en sus caminos. Después e seleccionar la recogida ya no aparece en la lista general, pero aparecerá en la lista de recogidas específica del usuario.

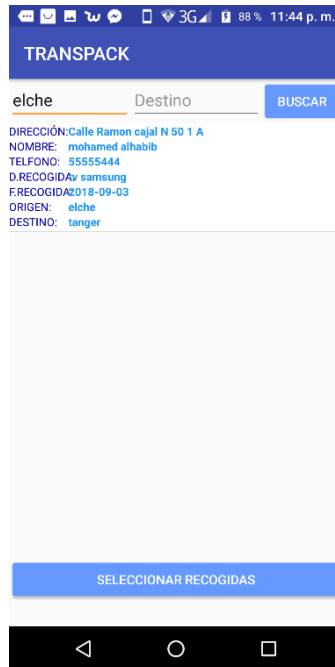


FIGURA 48: LISTA GENERAL DE RECOGIDAS

Registrar paquetes (repcionista en la oficina)

En la primera pantalla buscamos y seleccionamos el cliente, aparecerá la segunda pantalla donde metemos los datos del paquete pulsamos el botón registrar mercancía y aparece la tercera pantalla donde se genera el código QR del paquete que se usa después para añadir el paquete el viaje. Pero de momento solo se almacena en el almacén para su posible asignación a un viaje.

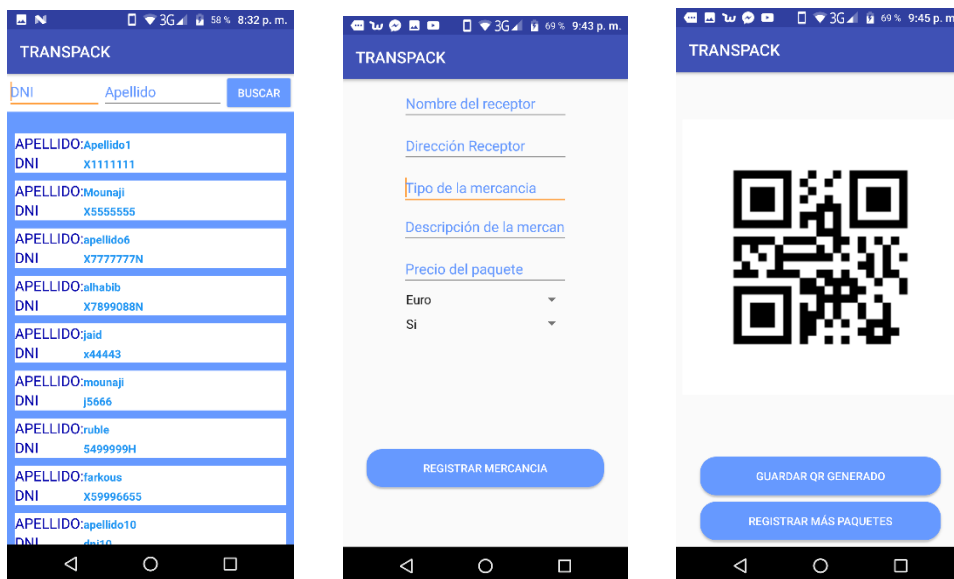


FIGURA 49: REGISTRO DE PAQUETES

Añadir paquete durante el viaje (Transportista)

Durante el viaje (en la empresa familiar) el transportista suele concretar un punto de encuentro donde se reúnen clientes y no clientes que quieren enviar sus paquetes a Alemania entonces lo que hace el transportista añadir estos paquetes al viaje registrando primero los clientes si no están registrados. En este caso, en el registro del paquete se asigna directamente el paquete al viaje (no hace falta escanear el código QR generado para asignarlo al viaje) y al mismo tiempo se genera el código QR.

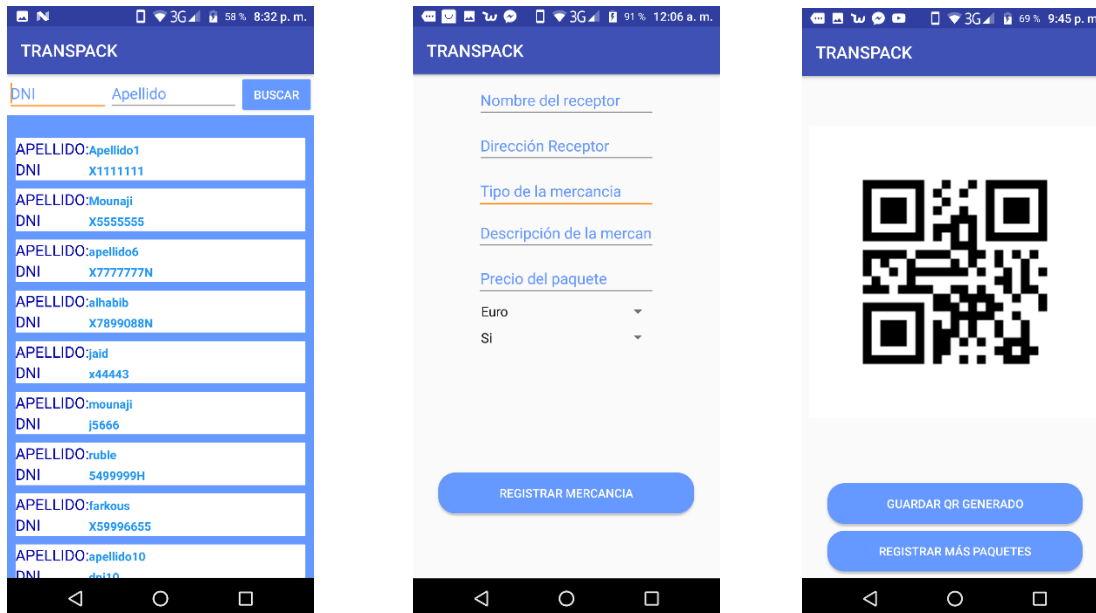


FIGURA 50: AÑADIR PAQUETE DURANTE EL VIAJE

Pasar paquete de un viaje a otro

Esta parte fue creada para obtener la opción de pasar un paquete de un viaje a otro por varios motivos por ejemplo que el primer transporte deja de funcionar o que el primer viaje ha terminado y lo pasa al siguiente viaje que llega hasta el destino. Primero se escanea el paquete seleccionamos el viaje que queremos que pase el paquete y listo.

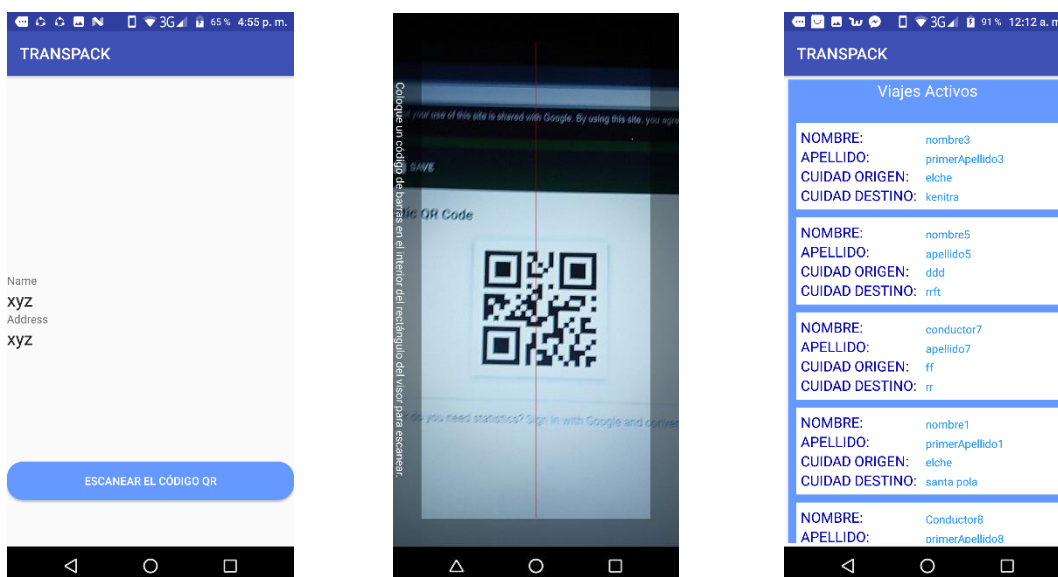


FIGURA 51: PASAR PAQUETE A OTRO VIAJE

Entregar paquete

Primero aparece la pantalla de clientes donde aparecen los que tienen paquetes a entregar en este viaje, seleccionamos un cliente luego aparece la segunda pantalla donde aparecen los paquetes del cliente a entregar. La entrega se realiza en un punto de encuentro donde todos los clientes se encuentran con el transportista y allí se realiza la operación.

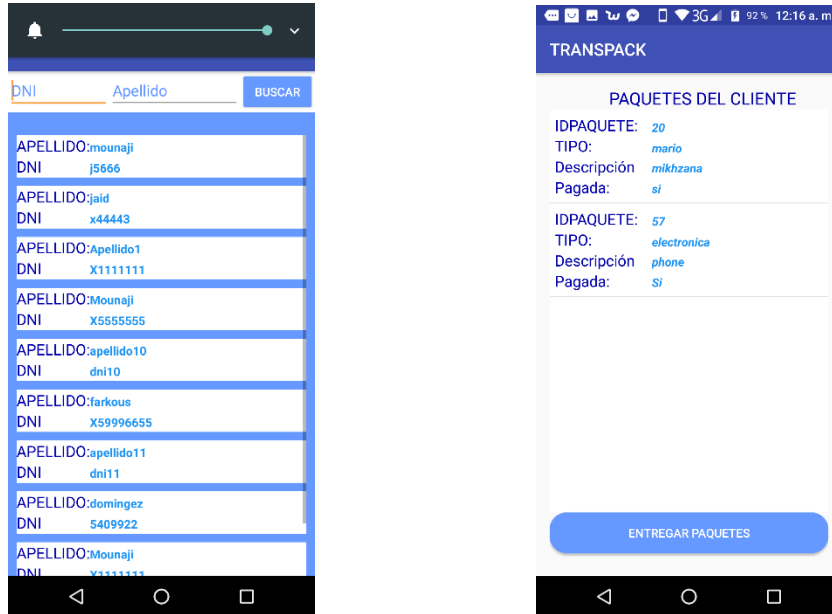


FIGURA 52: ENTREGA DE PAQUETES

Nueva incidencia

En la primera página buscamos y seleccionamos el cliente y nos aparece la segunda pantalla donde insertamos la incidencia y pulsamos el botón nueva incidencia

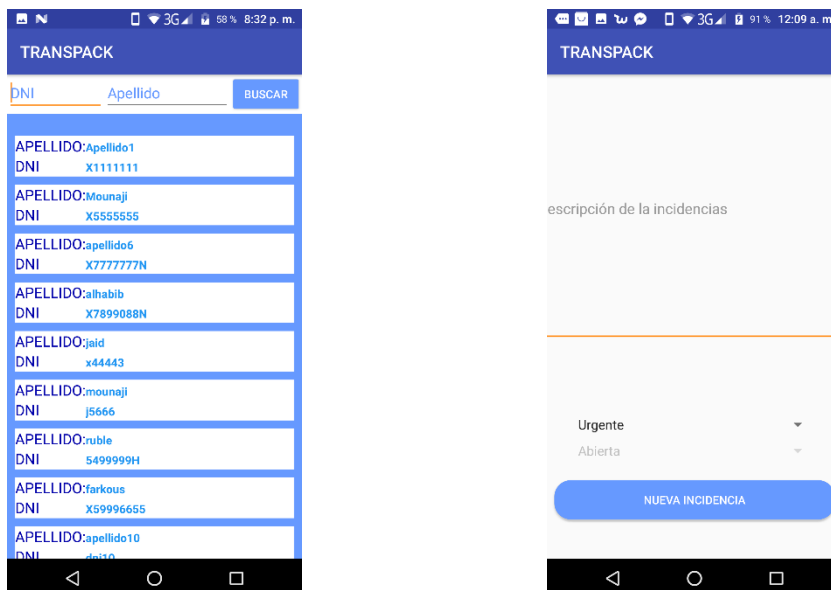


FIGURA 53: AÑADIR INCIDENCIA

Modificar transportista durante el viaje

En esta pantalla realizamos el cambio del transportista original del viaje por otro que no tiene ningún viaje asignado.

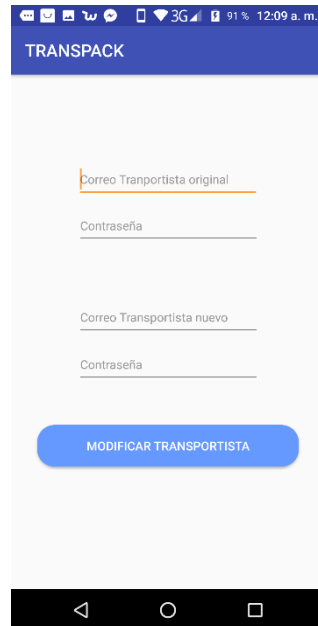


FIGURA 54: MODIFICAR TRANSPORTISTA

Crear viaje

En la primera pantalla el transportista crea el viaje rellenando los campos y pulsando el botón crear viaje, después aparecerá la segunda pantalla donde escañemos todos los paquetes (cuando se escanea un paquete en este caso se asigna al viaje)

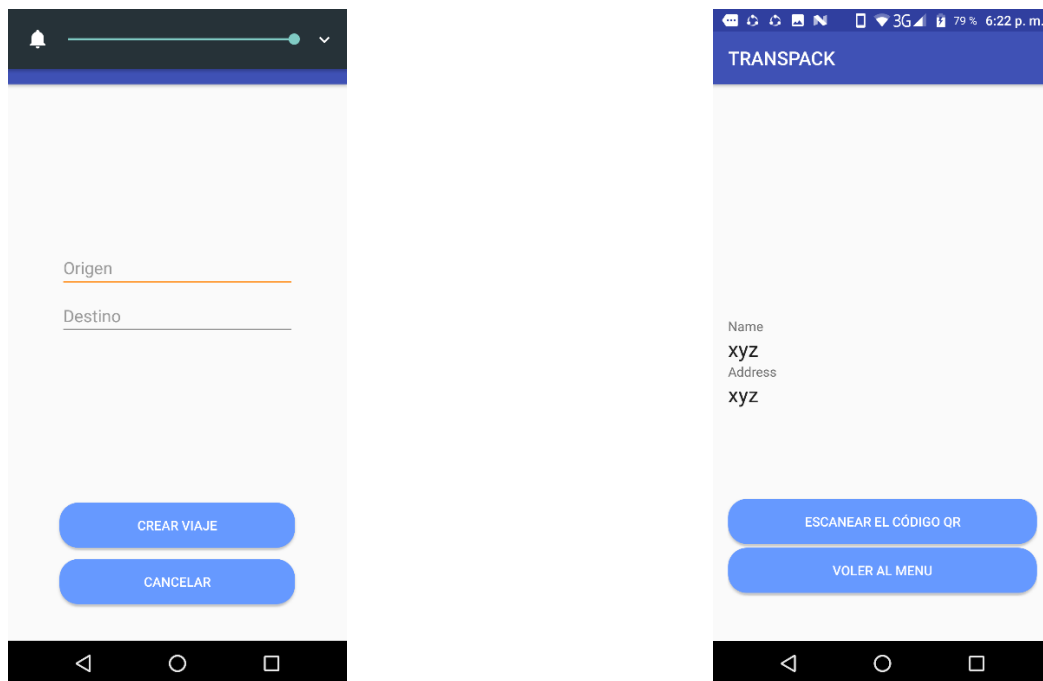


FIGURA 55: CREAR VIAJE

Cerrar viaje

En esta pantalla si el transportista tiene un viaje asignado puede cerrarlo, sino saldrá un mensaje indicando que no tiene viaje asignado para cerrarlo



FIGURA 56:CERRAR VIAJE

Añadir coste al viaje

En esta pantalla añadimos los costes que se gastan durante el viaje, añadiendo la cantidad del coste su descripción y la unidad monetaria con los viajes están entre Europa y Marruecos las unidades monetarias son en Euro y Dh.

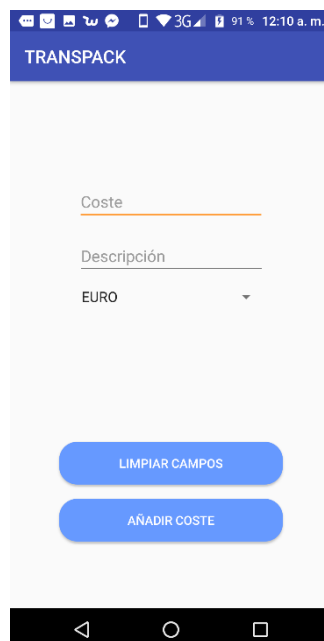


FIGURA 57: AÑADIR COSTE

Viajes activos

En esta pantalla aparecen todos los viajes activos.

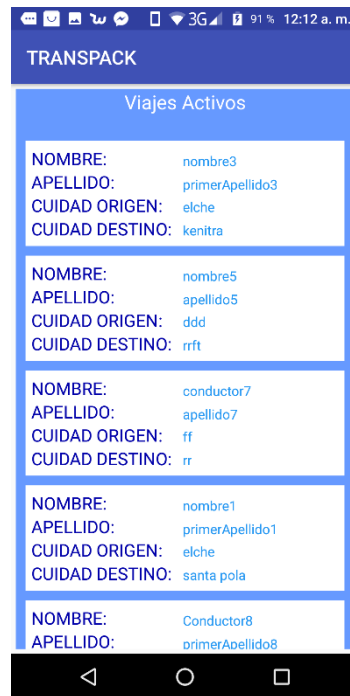


FIGURA 58: VIAJES ACTIVOS

Descubrir paquete

Con las dos primeras pantallas escaneamos el paquete, en la tercera aparecen los datos del paquete.

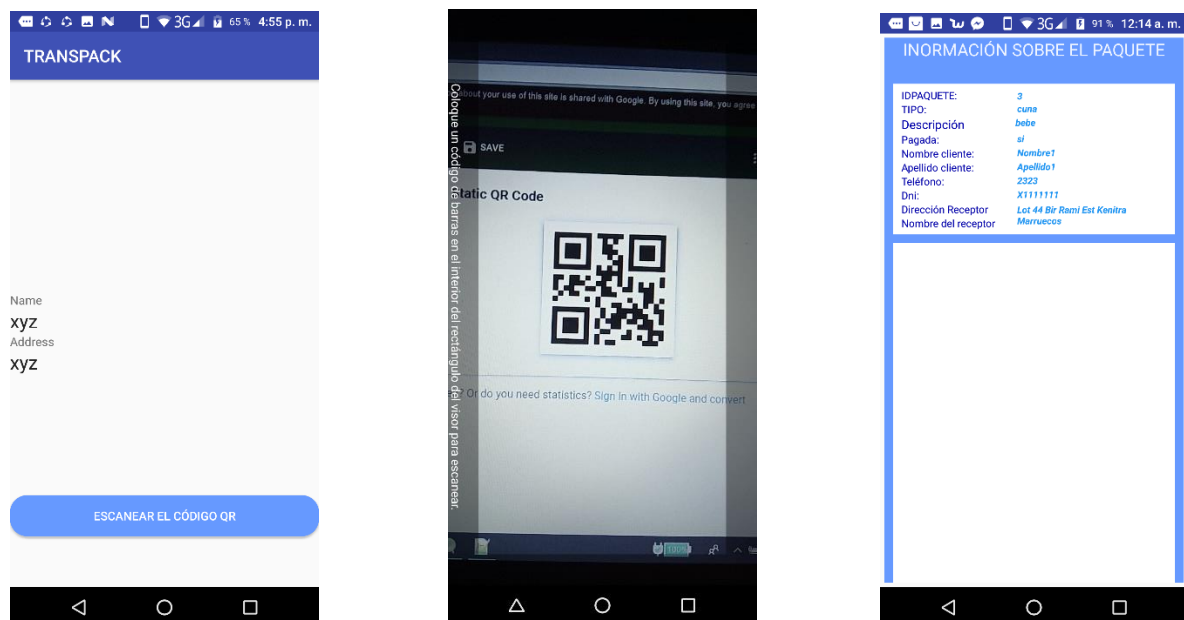


FIGURA 59: DESCUBRIR PAQUETE

Lista de entregas

En esta pantalla se muestran las entregas realizadas y la no realizadas, las primeras aparecen con color verde y las segundas con el color rojo.



FIGURA 60: LISTA ENTREGAS

Lista de incidencias

Esta pantalla muestra todas las incidencias creadas durante el viaje.

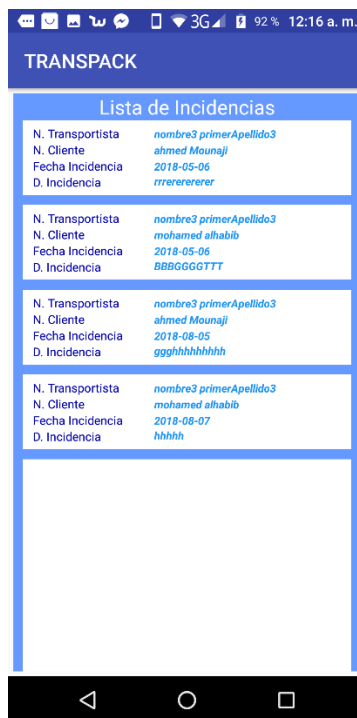


FIGURA 61: LISTA INCIDENCIAS

Lista de recogidas del viaje o el transportista

En la primera pantalla aparecen la lista de recogidas del transportista, buscamos y seleccionamos la recogida como resultado aparece la segunda pantalla donde aparecen los datos de la mercancía, pulsamos el botón registrar paquete, el resultado es paquete asignado al viaje con el código QR generado.

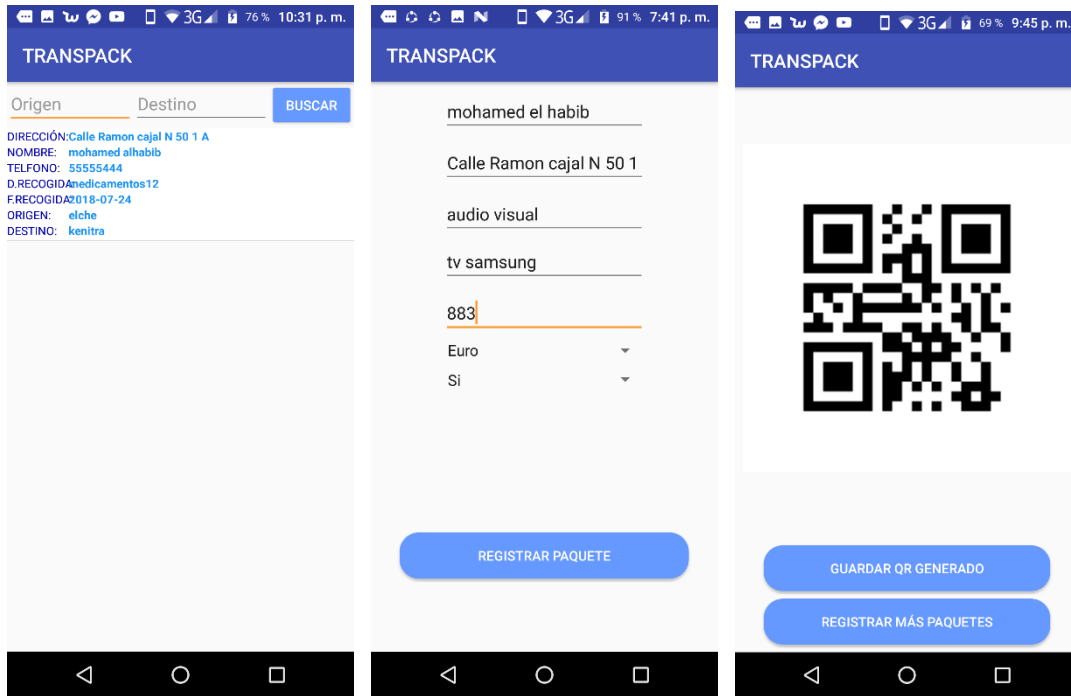


FIGURA 62: LISTA DE RECOGIDAS DEL TRANSPORTISTA

10. Pruebas

10.1 Introducción

A lo largo de este capítulo se detallarán el conjunto de las pruebas realizadas al software desarrollado. El objetivo que se pretende conseguir mediante la realización de este conjunto de pruebas es valorar la calidad del software, y además nos permitirá detectar y corregir los posibles errores que se produzcan.

La fase de pruebas es una de las más importantes dentro del ciclo de desarrollo de un sistema software, debido a que la garantía de calidad del producto depende enormemente de las pruebas que se realicen sobre el mismo. Las pruebas que se han realizado han sido las pruebas de la caja blanca (las cuales están realizadas desde un enfoque estructural) y las pruebas de la caja negra (realizadas desde un enfoque funcional). La creación de pruebas minuciosas y bien planteadas nos permite ahorrar mucho tiempo y esfuerzo.

El enfoque estructural o prueba de la caja blanca consiste en centrarse en la estructura interna (implementación) del software en función de la cual elegir los casos de prueba. En el enfoque estructural, la prueba ideal (exhaustiva) consiste en probar todos los posibles caminos de ejecución que pueden ser trazados a través de las instrucciones del código.

Por otro lado, en **el enfoque funcional o prueba de la caja negra** nos centraremos en el estudio de la especificación de cada una de las funciones, sus entradas y sus salidas, para obtener así los diferentes casos de prueba. En este caso, la prueba ideal consiste en probar todas las posibles entradas y salidas del sistema software.

La filosofía de trabajo para llevar a cabo las pruebas debe basarse en los siguientes puntos:

- Ejecutar el programa con la intención de encontrar un error.
- Establecer pruebas que presenten alta posibilidad de error.
- Encontrar errores no detectados hasta entonces.

Teniendo en cuenta que el objetivo principal de la prueba es descubrir un error, se han tomado las siguientes pautas a la hora de realizar las pruebas:

- Cada caso de prueba elegido debe definir el resultado de salida esperado.
- Se debe analizar a conciencia el resultado de cada prueba para detectar posibles indicios de defectos en otras partes del programa.
- En cada caso de prueba se deben incluir tantos los casos válidos y esperados de entrada como válidos e inesperados.
- No deben plantearse las pruebas suponiendo que prácticamente no haya defectos en el programa.

Las pruebas se han realizado desde un enfoque estructural (pruebas de la caja blanca) y desde un enfoque funcional (pruebas de la caja negra):

Las pruebas han sido realizadas durante la etapa de codificación del sistema software y tras su finalización y han sido llevadas a cabo por el autor del sistema y por personas ajenas al proceso de desarrollo. A continuación, se describirán de forma más detallada.

10.2 pruebas de la caja blanca o enfoque estructural

Este conjunto de pruebas ha sido realizado por el autor del trabajo de forma paralela a la etapa de codificación, es decir, al mismo tiempo que se iban implementando cada una de las funciones y procedimientos se iban realizando las pruebas necesarias sobre todos los componentes del software que se iban desarrollando para probar su validez. En cada caso se han realizado pruebas para probar todos los posibles caminos de ejecución del software, la no existencia de bucles infinitos, valores de los parámetros y de las variables, etc.

Los errores que se han ido encontrando mediante la ejecución de los diferentes caminos de ejecución que se formaban mediante las instrucciones del código de la aplicación, han sido corregidos y probados nuevamente para comprobar que estaban subsanados prestándose un interés especial a las consultas realizadas a la base de datos, comprobando que los resultados obtenidos eran los esperados.

Una vez terminadas las pruebas de caja blanca o enfoque estructural podemos afirmar que se han recorrido al menos una vez todos los posibles caminos independientes que se forman mediante las instrucciones del código de la aplicación y que han sido probados y corregidos en el caso de que hubiese errores durante la realización de este tipo de pruebas.

10.3 Pruebas de la caja negra o enfoque estructural

Las pruebas de la caja negra han sido realizadas posteriormente a la etapa de codificación. Para una mejor descripción de las pruebas se utiliza la tabla mostrada a continuación:

Precondiciones	
Acción	
Checkpoint	

TABLA 59: MODELO DE TABLA DE PRUEBAS

Se van a mostrar en seguida algunos ejemplos de las pruebas realizadas sobre toda la aplicación. Fue probado también el control realizado sobre el perfil de usuario, probando por ejemplo realizar operaciones que son permitidas solo por el manager con los otros usuarios (El transportista y el recepcionista) y las pruebas fueron satisfactorias, respondiendo siempre con los resultados esperados.

Precondiciones	<ul style="list-style-type: none">• El usuario debe estar registrado
Acción	<ul style="list-style-type: none">• El usuario introduce el correo y la contraseña• El usuario pulsa el botón Aceptar
Checkpoint	<ul style="list-style-type: none">• La aplicación valida los datos y muestra al usuario la pantalla del menú principal.

TABLA 60: LOGUEO CORRECTO DEL USUARIO

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar registrado
Acción	<ul style="list-style-type: none"> • El usuario introduce el correo y la contraseña • El usuario pulsa el botón Aceptar
Checkpoint	<ul style="list-style-type: none"> • La aplicación no valida los datos introducidos y muestra al usuario un mensaje de error.

TABLA 61: LOGUEO INCORRECTO DEL USUARIO

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar logueado • El usuario puede ser Manager / Transportista / Recepcionista
Acción	<ul style="list-style-type: none"> • El usuario introduce datos validos del cliente • El usuario pulsa el botón Aceptar
Checkpoint	<ul style="list-style-type: none"> • La aplicación valida los datos introducidos y muestra un mensaje de confirmación.

TABLA 62: REGISTRO UN CLIENTE CON DATOS VALIDOS

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar logueado
Acción	<ul style="list-style-type: none"> • El usuario no introduce todos los datos necesarios • El usuario pulsa el botón Aceptar
Checkpoint	<ul style="list-style-type: none"> • La aplicación no valida los datos introducidos y muestra al usuario un mensaje de advertencia para introducir todos los datos necesarios.

TABLA 63: REGISTRO DE UN CLIENTE CON DATOS NO VALIDOS

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar logueado
Acción	<ul style="list-style-type: none"> • El usuario selecciona el cliente de la lista • El usuario no introduce todos los datos necesarios • El usuario pulsa el botón Actualizar datos
Checkpoint	<ul style="list-style-type: none"> • La aplicación no valida los datos introducidos y muestra al usuario un mensaje de advertencia para introducir todos los datos necesarios.

TABLA 64: ACTUALIZAR LOS DATOS DE UN CLIENTE CON DATOS NO VÁLIDOS

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar logueado
Acción	<ul style="list-style-type: none"> • El usuario selecciona el cliente de la lista • El usuario introduce todos los datos necesarios • El usuario pulsa el botón Actualizar datos
Checkpoint	<ul style="list-style-type: none"> • La aplicación valida los datos introducidos y muestra al usuario un mensaje de éxito.

TABLA 65: ACTUALIZAR LOS DATOS DE UN CLIENTE CON DATOS VÁLIDOS

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar logueado
Acción	<ul style="list-style-type: none"> • El usuario filtra por Dni y/o apellido del cliente introduciendo datos correctos del cliente • El usuario pulsa el botón buscar
Checkpoint	<ul style="list-style-type: none"> • Aparece el cliente en la lista de búsqueda.

TABLA 66: BUSQUEDA DEL CLIENTE CON DATOS CORRECTOS

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar logueado
Acción	<ul style="list-style-type: none"> • El usuario filtra por Dni y/o apellido del cliente introduciendo datos incorrectos del cliente • El usuario pulsa el botón buscar
Checkpoint	<ul style="list-style-type: none"> • No aparece el cliente en la lista de búsqueda.

TABLA 67: BÚSQUEDA DEL CLIENTE CON DATOS INCORRECTOS

En el caso de que un usuario no fuera Manager, el sistema no le permite gestionar usuarios.

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar logueado • El usuario puede ser Manager
Acción	<ul style="list-style-type: none"> • El usuario introduce datos validos del usuario. • El usuario pulsa el botón Aceptar
Checkpoint	<ul style="list-style-type: none"> • La aplicación valida los datos introducidos y muestra un mensaje de confirmación.

TABLA 68: REGISTRO DE UN USUARIO CON DATOS VALIDOS

Precondiciones	<ul style="list-style-type: none"> • El usuario debe estar logueado
Acción	<ul style="list-style-type: none"> • El usuario no introduce todos los datos necesarios • El usuario pulsa el botón Aceptar
Checkpoint	<ul style="list-style-type: none"> • La aplicación no valida los datos introducidos y muestra al usuario un mensaje de advertencia para introducir todos los datos necesarios.

TABLA 69: REGISTRO DE UN USUARIO CON DATOS INVALIDOS

Precondiciones	<ul style="list-style-type: none"> • El Manager debe estar logueado
Acción	<ul style="list-style-type: none"> • El Manager selecciona un usuario de la lista • El Manager no introduce todos los datos necesarios • El Manager pulsa el botón Actualizar datos
Checkpoint	<ul style="list-style-type: none"> • La aplicación no valida los datos introducidos y muestra al usuario un mensaje de advertencia para introducir todos los datos necesarios.

TABLA 70: ACTUALIZACIÓN DE UN USUARIO CON DATOS NO VÁLIDOS

Bibliografía

Cuando se elabora una bibliografía se deben describir los documentos consultados redactando referencias bibliográficas. Se trata de facilitar el acceso futuro a los documentos originales haciendo constar los datos fundamentales de cada documento de manera sencilla pero normalizada. Así, en este apartado, se recopilará todos los recursos utilizados para la elaboración de un trabajo de estas características. Recogiendo todos los artículos, libros y fuentes necesarias para la creación de este Trabajo de Fin de Grado.

Sitios Web:

- [1] https://es.wikipedia.org/wiki/Historia_del_tel%C3%A9fono_m%C3%B3vil
- [2] <https://www.universidadviu.es/evolucion-la-red-comunicacion-movil-del-1g-al-5g/>
- [3] https://elpais.com/tecnologia/2018/02/27/actualidad/1519725291_071783.html
- [4] <https://es.ccm.net/faq/1223-teclado-qwerty-y-azerty>
- [5] <https://www.apple.com/es/ios/app-store/>
- [6] <https://play.google.com/store?hl=es>
- [7] <https://desarrolloweb.com/articulos/que-es-webkit.html>
- [8] https://www.efecto2000.es/blog/10_la-evolucion-de-del-sistema-operativo-android.html
- [9] https://www.efecto2000.es/blog/10_la-evolucion-de-del-sistema-operativo-android.html
- [10] <https://developer.android.com/guide/platform/?hl=es-419#art>
- [11] <https://source.android.com/setup/build/jack?hl=es-419>
- [12] <https://developer.android.com/guide/platform/j8-jack?hl=es-419>
- [13] <https://developer.android.com/guide/components/services?hl=es-419>
- [14] <https://developer.android.com/guide/components/activities?hl=es-419>
- [15] <https://developer.android.com/reference/android/view/ViewGroup>
- [16] <https://developer.android.com/guide/topics/resources/overview?hl=es-419>
- [17] <https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419>
- [18] <http://siul02.si.ehu.es/~jimena/ABD/fuentes/ClienteServidor.pdf>
- [19] <https://es.wikipedia.org/wiki/LAMP>
- [20] <https://es.wikipedia.org/wiki/Webmin>
- [21] <https://es.wikipedia.org/wiki/PhpMyAdmin>

Libros

[22] Version control with Git

[23] Android Application Development Cookbook 2016 Rick Boyer, Kyle Mew Packt Publishing

[24] Android Apps for Absolute Beginners 2011 Wallace Jackson

[25] Administración de bases de datos Michael V.Mannino

[26] Sistemas de gestores de bases de datos María Jesús Ramos, Alicia Ramos, Fernando Montero Mc Graw Hill Mariano García Díaz 2006

[27] Análisis y Diseño de Sistemas, Kendal & Kendal Pearson 2011

[28] David Arnow, Gerald Weiss. "Introducción al a programación con JAVA. Un enfoque orientado a objetos". Madrid. Pearson Educación S.A. 2001.

El presente Trabajo Fin de Grado describe el desarrollo de un producto software, concretamente, de una aplicación móvil en el sistema operativo Android que se dedica a la gestión y trazabilidad de transporte de mercancías y paquetes para una empresa familiar entre Europa y Marruecos.

Tras realizar varias reuniones con el cliente, he llegado a entender sus necesidades en cuanto al proceso de transporte de la empresa, lo que me ayudo a diseñar una aplicación que se adapta a los requerimientos del cliente y cumpliendo con los objetivos planificados.