Universidad Zaragoza
1542

Trabajo Fin de Grado

App para la carga de trenes militares: lectura automatizada de materiales

Autor

Patricia Díaz Aguilar

Director/es

Director académico: Dr. D. Carlos Borau Zamora

Director militar: CAP D. Óscar Luis Gálvez Cortés

Centro Universitario de la Defensa-Academia General Militar

Año 2019-2020

En agradecimiento a los Oficiales, Suboficiales y Tropa de la Compañía de Ferrocarriles y del Centro de Control de Movimientos por Ferrocarril del Regimiento de Pontoneros y Especialidades Nº12, así como a los tutores directores de este proyecto y todo aquel personal que se haya visto involucrado en la realización del mismo, por su ofrecimiento y dedicación para llevar a cabo este trabajo.

Índice

Lista de figuras

Lista de tablas

Lista de abreviaturas

UE: Unión Europea

OTAN: Organización del Tratado del Atlántico Norte

TEN-T: Red Transeuropea de Transporte

CCMR: Célula de Control de Movimiento por Ferrocarril

FFCC: Ferrocarriles

ROC: Responsable de Operaciones de Carga

RENFE: Red Nacional de Ferrocarriles Españoles

ADIF: Administrador de Infraestructuras Ferroviarias

UCO: Unidad Central Operativa

ET: Ejército de Tierra

BESP: Batallón de Especialidades

MALE: Mando de Apoyo Logístico del Ejército

MING: Mando de Ingenieros

MINISDEF: Ministerio de Defensa

TO: Teatro de Operaciones

ZO: Zona de Operaciones

SIGLE: Sistema Integrado de Logística del Ejército

SUBGES: Subdirección de Gestión

PAP-T: Programación Anual

PT: Petición de transporte

IDC: International Data Corporation

RPEI12: Regimiento de Pontoneros y Especialidades Nº12

QR: Quick Response

UML: Unified Modeling Language

1. Resumen

El transporte militar por ferrocarril es una de las ramas más específicas de la especialidad fundamental de Ingenieros, siendo uno de los medios más eficaces para transportar tanto mercancía como personal. Para enfocar este proyecto, se ha centrado en el proceso del transporte ferroviario, el cual engloba numerosos documentos a elaborar previamente por el CCMR (Célula de Control de Movimiento por Ferrocarril), llegando al informe final del tren cargado en cuestión, siendo este el que genera la necesidad de la aplicación desarrollada con este proyecto.

Siguiendo esta línea, dentro de este proceso intervienen numerosos actores, entre los cuales cabe destacar el papel que desempeña el Responsable de Operaciones de Carga (ROC), quien ejerce como cargador durante el proceso de embarque de material en el tren. La tarea final que ejecutar por este es identificar (a simple vista), cargar y registrar manualmente todo aquel material que ocupa cada determinada plataforma en la composición del tren. Con el fin de agilizar esta tarea y disminuir en gran medida el error humano, se ha desarrollado durante el periodo de prácticas externas, una aplicación móvil para sistemas operativos Android, la cual se encarga de automatizar este trabajoso proceso mediante la lectura de códigos QR. Además, ha sido probada en un transporte real y ha cumplido satisfactoriamente sus objetivos, encontrándose actualmente en estudio de implantación en el proceso de carga y a la espera de ser homologada por los organismos competentes.

ABSTRACT

Military rail transport is one of the most specific branches of the fundamental specialty of Engineers, being one of the most efficient means of transporting both goods and personnel. In order to focus this project, it has focused on the railway transport process, which includes numerous documents to be previously elaborated by the CCMR (Railway Movement Control Cell), arriving at the final report of the train loaded in question, being this the one that generates the need of the application developed with this project.

Along these lines, many actors are involved in this process, including the role played by the Cargo Operations Manager, who acts as loader during the process of loading material onto the train. The final task to be performed by him is to identify (at a glance), load and manually register all the material that occupies each particular platform in the composition of the train. In order to speed up this task and greatly reduce human error, a mobile application for Android operating systems has been developed during the internship period, which automates this laborious process by reading QR codes. In addition, it has been tested in a real transport and has satisfactorily met its objectives, and is currently being studied for implementation in the loading process and awaiting approval by the competent bodies.

1. Introducción

En esta memoria se va a desarrollar el Trabajo de Fin de Grado con título *App para la carga de trenes militares: lectura automatizada de materiales,* realizado durante el periodo de prácticas externas, las cuales han sido llevadas a cabo en el Regimiento de Pontoneros y Especialidades nº12 (RPEI12).

1.1.Alcance y ámbito de aplicación

La presente existencia de una amenaza en el flanco este de la OTAN ha convertido al ferrocarril en un transporte estratégico de primer nivel, originando la necesidad de proyectar y transportar los medios y, principalmente, los vehículos pesados hacia las zonas fronterizas, adaptándose así al escenario híbrido del conflicto[1] al que nos enfrentamos actualmente.

Es por esto por lo que la OTAN pide mayor coordinación entre los países aliados, a la vez que muestra su "profunda preocupación" por el aumento de la capacidad militar de Rusia y sus constantes esfuerzos para desestabilizar a países de la región[1]. Así es como el transporte ferroviario ha adquirido vital importancia en territorio internacional tratando de poner fin a las tensiones en focos como Europa, Siria o Corea.

Por otro lado, la UE lleva a cabo un plan de acción sobre la movilidad militar, tratando de identificar la Red Transeuropea de Transporte (TEN-T), mejorar las infraestructuras internacionales, invertir en el desarrollo de la dualidad de los trenes (transporte de personal y mercancía), y tratar de simplificar trámites aduaneros y los procedimientos del paso de fronteras. Todo esto se debe a la evolución del escenario de conflicto, el cual engloba todos los espectros del combate. [2]

Ilustración 1.Transportes internacionales el pasado año 2018.

[1] En la *Ilustración 2* se refleja dicho "escenario híbrido". Dicha imagen se ha obtenido de la aplicación LifeUAMap, la cual muestra en tiempo real todo tipo de conflicto de carácter militar en todas las regiones del mundo y cada uno de estos viene representado por un icono con forma de circulo que determina el tipo de acción. (p. ej: las acciones aéreas se corresponden el icono de un avión)

Ilustración 2. Escenario híbrido del conflicto. Aplicación LifeUAMap.

A esto se le añade el desarrollo tecnológico de las últimas décadas, que ha posicionado a las apps móviles como una de las herramientas más eficaces para optimizar tiempos en multitud de escenarios. De esta forma, la Compañía de Ferrocarriles ha optado por lanzar la propuesta del desarrollo de una aplicación que ayude a agilizar el proceso de transporte ferroviario.

Esta aplicación va dirigida a todos los órganos y personal del Regimiento que intervienen en el transporte por ferrocarril (definidos en la NAI2403/17[3]),especialmente para los responsables de operaciones de carga, quienes son los encargados de registrar todo aquel material que es transportado por este medio y las plataformas donde se carga; y para el personal del CCMR, cuya misión es coordinar los transportes por ferrocarril que realizan las distintas Unidades, realizando la planificación y seguimiento de los movimientos y actuando de intermediarios entre la empresa ferroviaria (RENFE) y las Unidades transportadas. Para ello, se ha dispuesto de las siete semanas de prácticas externas para el estudio, la creación y el desarrollo de esta.

1.2.Objetivos y requisitos

En el caso de la definición de los objetivos, es muy importante reconocer qué problemas va a resolver la aplicación. A continuación, se describen los inconvenientes que se presentan durante el proceso de carga junto con los objetivos del proyecto y la finalidad que se persigue con el alcance de estos:

- Principalmente, la creación de una aplicación de este tipo trata de simplificar al máximo el proceso de carga. El problema actual a la hora de realizar el embarque del material es que abarca un trabajo manual muy laborioso y, de esta forma, se facilitaría el procedimiento quedando registrado el qué, dónde, cuándo y el orden en el que se ha ido cargando el tren en un plazo de tiempo mucho más reducido.
- Por otro lado, debido a las reducidas variaciones en los distintos modelos de cada tipo de vehículo/material (en cuanto a peso y dimensiones), la aplicación trata de evitar equivocaciones humanas a la hora de reconocer el material que se está cargando y

solventar los cambios de última hora en la configuración de las hojas de carga resultantes.

- Asimismo, la aplicación llevará a cabo el cálculo de la tasación para la facturación del transporte, conforme a las tarifas que aplica la operadora *Renfe* en función del peso del tren y la distancia del trayecto recorrido, que serán explicadas posteriormente.
- En definitiva, se trata de una aplicación para la comprobación de los cuadros de composición y las peticiones de transporte planificados previamente en un tiempo mucho más reducido, dejando como resultado un proceso de carga más eficaz y, en caso de incidencias como la no correspondencia con la hoja de carga ya programada, esta sea actualizada en una franja de tiempo más corta.

Por otro lado, para definir los requisitos es necesario saber cuáles son las especificaciones requeridas por el personal al que va destinado su uso, resultando los siguientes requisitos:

- Que la aplicación que sea de fácil uso y manejo.
- Que mediante la lectura de un código QR único asociado a cada material, permita reconocer de forma inmediata el material exacto que se embarca en el tren.
- Que permita, a modo informativo, la visualización de las distintas fichas de gálibo[2] y los planos con las medidas del material, donde se detalla el ajuste del cargamento en función de sus propias dimensiones y de la plataforma donde se cargan (destinado principalmente para el personal con escasa experiencia). Se adjunta un ejemplo de una de estas fichas en el **ANEXO A.**
- Por último, que se programe de tal forma que sea fácilmente legible y quede abierta a futuras modificaciones a causa de la constante actualización de las normativas que se aplican y los recurrentes cambios de software/hardware de los dispositivos móviles.

Asimismo, se aportará un manual de usuario que explique en detalle cómo manejar la aplicación y en caso de ser necesario, como modificar las bases de datos conforme las actualizaciones que surgen constantemente respecto al material autorizado a ser cargado.

1.3. Metodología y estructura de la memoria

1. Consulta información general:
   a. Marco legal y documentación aplicable: convenio MINISDEF-RENFE, consigna C-41, catálogos de vagones, NC-007-07-17, etc.
   b. Estructura y funcionamiento del Regimiento de Pontoneros y Especialidades nº12 (RPEI12)
   c. Consulta de programas ya existentes con una finalidad similar, así como programas para el desarrollo de la aplicación y el lenguaje de programación.

[2] El **gálibo** es un conjunto de datos destinados a definir las condiciones dimensionales de circulación del material rodante regulado en Renfe para la Instrucción General N.º 66 [21] actualizada. El gálibo de cargamento se compone de un contorno de referencia corresponde a una sección tomada en línea recta y una de las reglas que hay que aplicar para tener en cuenta los diversos elementos de desplazamiento de la curva.[22]

2. Consulta información específica a los encargados del transporte por ferrocarril. Con la finalidad de establecer los objetivos y requisitos que deben cumplirse con este proyecto, se han llevado a cabo una serie de entrevistas con el personal que resulta más experimentado en este tipo de escenarios.
3. Estudio de la configuración de los ferrocarriles. Se analiza la infraestructura ferroviaria de la unidad, la composición de las distintas plataformas de carga y la configuración de los trenes, el material que está permitido cargar en las mismas, (regulado en las distintas normativas) y las tarifas correspondientes que se aplican durante el proceso de transporte.
4. Creación y desarrollo de la app. Se procede a la creación de una aplicación para dispositivos Android que, mediante la lectura de un código QR asociado a los distintos materiales regulados en el convenio MINISDEF-RENFE, permita reconocer dicho material en el mismo instante que se embarca y, una vez finalizado el cargamento de las plataformas, genera una hoja de carga en formato .csv, donde además, también consta el importe total de la facturación junto con el resto de datos del embarque (peso total del tren, plataformas, material embarcado, etc.).
5. Comprobación y puesta en escena. El día 10 de octubre se realizó un transporte por ferrocarril solicitado por la Brigada “Guzmán el Bueno” X por el itinerario Córdoba-El Higuerón/ San Gregorio, en el que se hizo una puesta en escena de la aplicación para su posterior evaluación y cuyo feedback se plasmará en el apartado 6 del presente trabajo.
6. Conclusiones. Se define el alcance de los objetivos previamente citados, así como las incidencias y los inconvenientes que se han planteado durante el mismo proceso de desarrollo de la aplicación**.**
7. Líneas futuras. Apartado donde se plantean una serie de trabajos futuros que ha dado lugar este proyecto, añadiendo distintas funciones y hacer más amplio el uso de la aplicación.

2. Transporte por ferrocarril.

Ilustración 3. Tren cargado del transporte de la BRIX.

Un subsistema de transporte es aquel instrumento mediante el cual el jefe del Mando de Apoyo Logístico (MALE), a su nivel, transporta los recursos de personal, ganado y material que el Ejército de Tierra (ET) necesita, empleando con la máxima eficiencia para situarlos con oportunidad y precisión en el tiempo y lugar ordenados y en las condiciones de empleo necesarias para el cumplimiento de las misiones. [4]

En este caso, entre dichos instrumentos, se encuentran los trenes militares. Se trata de trenes para uso exclusivo del Ejército, organizados normalmente con los medios de las compañías ferroviarias. Su composición, recorrido y horario son consecuencia de las necesidades específicas de transporte de aquellas unidades que lo solicitan. Estos pueden ser destinados para el transporte del personal, material o trenes mixtos (personal, material y/o ganado). [5]

2.1.Compañía de Ferrocarriles.

El Regimiento de Pontoneros y especialidades nº12 se encuentra encuadrado dentro del Mando de Ingenieros (MING) prestando capacidades con carácter de apoyo al combate. Las actividades que se desempeñan exigen un alto grado de formación, nivel técnico y experiencia y, entre todas estas actividades que desarrollan las unidades de especialidades, encontramos la capacidad de apoyo a la movilidad mediante la reconstrucción y habilitación de vías de comunicación, la capacidad de restablecimiento de vías de FF.CC. y la gestión trenes y operaciones de terminal; las cuales son competencia de la Cía. de ferrocarriles junto con la responsabilidad de todo el material rodante perteneciente a MINISDEF (Ministerio de Defensa). Esta es la disposición actual de la compañía:

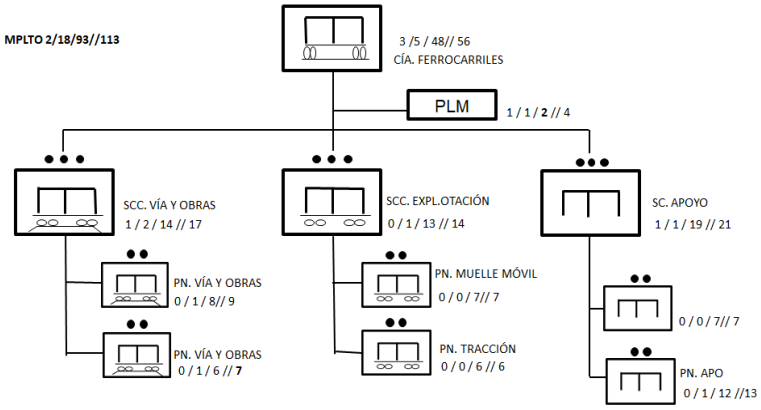


Ilustración 4.Organigrama actual de la Cía. de Ferrocarriles. Elaborado por el jefe de la Cía. de FFCC.

Además, los cometidos que se le asignan ejecutar a esta unidad son los siguientes:

- Formación de las composiciones de los trenes cuando se va a realizar un transporte, conforme a la Propuesta de Organización de Trenes remitida por el CCMR.
- Construcción, rehabilitación y mantenimiento de las instalaciones de la terminal y ramales de acceso.
- Realizar el transporte táctico en TO/ZO de unidades militares, formando una composición ferroviaria autónoma y de la forma más eficiente.
- Reconocimientos técnicos de terminales e infraestructura ferroviaria.

Es decir, dentro de las capacidades que presenta la unidad, son aptos a la hora de constituir, activar y explotar una Unidad Terminal tanto en Territorio Nacional como en Zona de Operaciones, incluyendo el embarque y desembarque de los medios transportados, tanto como la expedición e inspección de los trenes militares.

2.2.Célula de Control de Movimiento por Ferrocarril (CCMR)

A continuación, se va a explicar el papel que desarrolla el CCMR dentro del transporte por ferrocarril. Esta es Célula de Control de Movimiento por Ferrocarril y se encuentra en la Plana Mayor del Batallón de Especialidades. Este ha sido el órgano de asesoramiento más valorado en el desempeño de este proyecto.

Se trata de un subsistema de transporte que, con carácter general, es un órgano que se dedica a asesorar e informar en todo el espectro que abarca su especialidad, tanto sus capacidades como limitaciones dentro de la función logística de transporte; colaboran en la ejecución de los transportes y están en constante estudio sobre sobre las normas que puedan incluirse o modificarse dentro de lo que afecte a su especialidad.

El CCMR se articula en dos áreas a la hora de determinar sus cometidos y actúan sucesivamente. En primer lugar, tenemos el área de transporte la cual se encarga de los siguientes cometidos: [7] [6]

- Una vez elaborado el Programa Anual de Transporte Terrestre (PAP-T), el CCMR realiza una valoración de las posibles composiciones para los transportes que se establecen en el mismo, elaboran la propuesta organización de trenes y confeccionan los cuadros de composición.
- Coordinan el transporte tanto con las unidades a transportar como con Renfe.
- Asesorar a la UCO a transportar, en procedimientos, amarres, etc., proporcionándole la documentación necesaria y las fichas de gálibo correspondientes.
- Recabar de EF (RENFE) el correspondiente programa de transporte y prescripciones técnicas y remitirlo a SUBGES y Unidades a transportar.
- Tramitar los pasaportes y las comisiones del personal que participa en los transportes.
- Coordinar con Renfe la consignación y posicionamiento del material. Así como, cargar en el SIGLE (Sistema Integral de Gestión Logística del Ejército) todo el movimiento del material cuando se dirige a talleres para el mantenimiento de este.
- Realizar la facturación de los trenes a transportar.

Por otro lado, está el área de seguimiento, encargada de los siguientes cometidos:

- Preparación de la documentación del personal de embarque y acompañamiento.
- Impartir las instrucciones particulares y de coordinación, con la UCO y Renfe, al personal de circulación, embarque y acompañamiento, actuando como intermediario a lo largo de todo el proceso
- Operar los sistemas informáticos de seguimiento y establecer contacto telefónico con personal de circulación, embarque y acompañamiento y, de esta forma, estar constantemente informado respecto la situación y el estado del transporte.
- En caso de que se originen, poner en conocimiento de EF (RENFE) las incidencias.
- Recopilar toda la documentación generada y las novedades de desperfectos en el material, propiedad de MINISDEF, empleado en los transportes y comunicarlo a la Cía. FFCC, una vez finalice el transporte.

2.2.1. Documentación, programación y planificación.

En este proceso intervienen cuatro actores: la unidad transportada, la empresa *RENFE*, la Cía. de Ferrocarriles y el CCMR, quien actúa como intermediario entre todos ellos.

A lo largo de este proceso se recopilan numerosos documentos que engloban la programación y planificación de este [7][8]. Anualmente las distintas unidades elaboran el Programa Anual de Transporte Terrestre (PAP-T), donde se prevén todos los movimientos que se pretenden realizar por las unidades, ya sea por carretera o por ferrocarril; el CCMR elabora el cuadro de composición con la Propuesta de Organización de los Trenes en función del material que vaya a ser cargado; posteriormente, se realiza la Petición de Transporte (PT), donde se recoge el orden de las plataformas que conforman el tren y el orden de embarque del material en las mismas; *RENFE* envía el Programa de transporte, junto con el orden de marcha del tren con sus respectivas horarios y paradas y, por último, se genera el cuadro de carga, siendo este objetivo perseguido por la aplicación.

En este último se refleja, en orden de cabeza a cola del tren, el código identificativo y tipo de plataforma utilizado, y la clase y el peso del material embarcado en la misma, así como, que vehículos se ven afectados por la Consigna serie C-41[9]. Esta es la hoja resultante donde queda registrada toda la información del tren que, en un principio, debería ser equivalente a la PT, pero puede sufrir modificaciones dado que siempre hay que prever la posibilidad de tener que variar el orden de embarque de los vehículos, en función de la composición de las plataformas. En el **ANEXO B** se plasman los documentos tipo que se recogen a lo largo de este proceso.

No obstante, cabe destacar que estos son los documentos principales que se recogen a largo de la organización del transporte. Existe además una cantidad considerable de pasos intermedios que involucran en repetidas ocasiones a la Subdirección de Gestión (SUBGES), Renfe, las UCO trasportadas, y a los distintos órganos del Regimiento en cuestiones de conformidad con la programación que se establece.

Además, una vez que ya está todo embarcado en sus plataformas correspondientes, se lleva a cabo la tasación del tren, la cual también calculará la aplicación conforme a las tarifas establecidas con la empresa ferroviaria, que vendrán explicadas posteriormente.

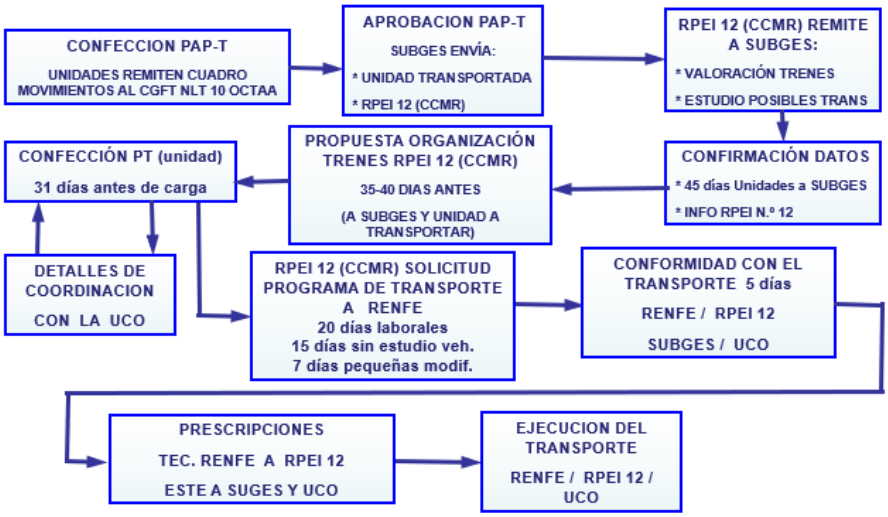


Ilustración 5. Proceso de transporte por ferrocarril. Elaboración propia.

2.3. Estudio de las plataformas de carga.

Las plataformas son las superficies físicas donde se embarca todo aquel material que va a ser transportado. Dentro de la composición de los trenes, existe una gran variedad de modelos de plataformas que varían en función del uso al que están destinadas, ya que este medio se puede transportar diversos vehículos sobre ruedas o cadenas, materiales a granel (en tolvas), material de artillería, así como personal y/o ganado, etc.

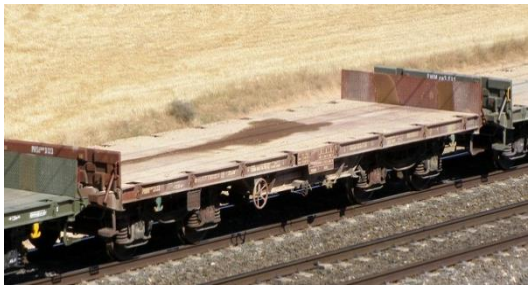
Las que se han estudiado para la realización de la aplicación son las siguientes, ya que son las más comúnmente usadas. Sin embargo, en el **ANEXO C** se adjuntan todos los tipos de vagones recogidos por el Manual Técnico del Transporte por Ferrocarril (MT5_007) [10], donde se reflejan los detalles técnicos de cada uno de ellos**.**

- **PMME y PMMR.** Estas son las plataformas pertenecientes a las unidades militares y son las que se usan normalmente en el transporte militar. Ambas son aparentemente iguales, su superficie es metálica y tienen las mismas dimensiones, pero, a diferencia de las PMME, las PMMER son plataformas reforzadas, es decir, que soportan mayor pesaje, y la distancia entre sus ejes es menor y soportan mayor carga. Actualmente el ET dispone de 45 y 55 respectivamente y todas ellas están identificadas mediante códigos denominados *UIC*, y se encuentran en la estación de ferrocarril ubicada en San Gregorio.
- **MM2.** Estas plataformas son de madera y pertenecen a la empresa Renfe. Son alquiladas para numerosos transportes por ferrocarril y principalmente son ocupadas por los vehículos de ruedas.
- **A10X y B10X**. Ambos dos son coches de viajeros, pero la distinción entre ellos es que los primeros corresponden a coches de 1ª clase y los segundos de 2ª. En estos ejercicios se usan para transportar al personal de escolta del tren, los auxiliares y los jefes de vehículos embarcados. Por lo tanto, debe de haber al menos uno por cada tren obligatoriamente.

- **Muelles <<TRANSFER>> (PM).** Se trata de un muelle testero[3] transportable estas plataformas llevan incorporado el muelle de carga que permiten el embarque tanto frontal como lateral cuando no se dispone de playas de embarque[4] en la propia estación.

COCHE VIAJEROS

PMME/PMMER



MM2

MUELLE TRANSFER

Ilustración 6. Plataformas de carga.

2.4.Material de cargamento.

El material que está permitido cargar en estos trenes viene regulado por un convenio entre el Ministerio de Defensa y la Operadora RENFE para el transporte de mercancías y personal [11]. Entre estos, podemos encontrar diversos tipos de mercancía como carros de combate, vehículos blindados, vehículos automóviles, material de artillería, maquinaria, remolques y/o semirremolques.

Además, debemos tener en cuenta que numerosos de los diseños de estos vehículos y/o material se acogen a una serie de restricciones recogidas en la CONSIGNA C-41. Dicha consigna expone las normas que deben cumplir los vehículos militares con características excepcionales y se trata de la recopilación de una serie de apéndices que afectan a vehículos específicos, los cuales muestran unas limitaciones por exceso estudiadas por ADIF.

3 El **testero** es la parte frontal de un tren o ferrocarril.
4 Las **Playas de embarque** son las zonas habilitadas para cargar los trenes dentro de una estación.

Por un lado, es necesario acondicionar el gálibo a los materiales que sufren el exceso del mismo. Es por ello que la citada consigna recoge planos detallados sobre las condiciones particulares que deben cumplir los materiales presentes en el documento en función del tipo de plataforma donde serán cargados: estas son las fichas de gálibo. Además, quedan redactadas tanto las disposiciones generales como particulares de circulación donde se limita la velocidad de los trenes en función de los itinerarios, estaciones o vías que se transiten con dicho material embarcado.

Por otro lado, es de tener en cuenta que todo material requiere un estudio detallado para ver si es viable su transporte por vía férrea. Este se realiza por parte del CCMR con el objetivo de analizar si es preciso tratarlo como "mercancía excepcional"; lo que sucede cuando el material o el cargamento exceden los límites de gálibo de cargamento establecidos.

2.5.Tarifas.

En el convenio citado previamente [11], se establecen de la misma forma las tarifas que se aplican en la facturación del proceso de transporte por ferrocarril. Estas están acordadas dependiendo del tipo de plataforma donde se transporta la carga, el tipo de material que las ocupan y su peso correspondiente, y la distancia recorrida, la cual ya está fijada para cada itinerario en concreto.

- Los itinerarios autorizados están previamente definidos en la consiga C-41 y se trata de 16 trayectos cuya distancia ya está fijada a la hora de facturar el tren. También se pueden realizar trayectos como combinación de las paradas intermedias en los distintos itinerarios. [12]

ORIGEN	***DESTINO***	***DISTANCIA(KMS)***
Badajoz	San Gregorio	967
Badajoz	Chinchilla	601
Benahadux	San Gregorio	967
Chinchilla	Córdoba-El Higuerón	445
Chinchilla	El Goloso	322
Chinchilla	Valencia Fte. S. Luis	182
Córdoba-El Higuerón	San Gregorio	816
El Goloso	San Gregorio	387
Júndiz	San Gregorio	303
Ronda	San Gregorio	1012
San Gregorio	Valencia Fte. S. Luis	543
San Gregorio	Valladolid Argales	629
San Gregorio	Villafría (Burgos)	355
Tarragona	San Gregorio	285
Júndiz	Chinchilla	702
Valladolid Argales	Chinchilla	545

Tabla 1. Itinerarios autorizados para los transportes. Elaboración propia con los itinerarios de la consigna.[11]

- Las tarifas que se aplican son las siguientes:

Material ferroviario propiedad de Renfe		
TIPO DE MATERIAL	**PESO**	**TARIFA**
Vehículos blindados	>= 15 Tm	0,1215€/Tm
Vehículos blindados	< 15 Tm	0,1509€/Tm
Resto de material	-	0,1931€/Tm
Transporte internacional	-	0,1931€/Tm
Material ferroviario propiedad de MINISDEF		
TIPO DE MATERIAL	**PESO**	**TARIFA**
Vehículos blindados	>= 15 Tm	0,1057€/Tm
Vehículos blindados	< 15 Tm	0,1298€/Tm
Resto de material	-	0,1660€/Tm
Transporte internacional	-	0,1660€/Tm
Coches de viajeros propiedad de MINISDEF (en trenes de mercancías)		
Coche de viajeros		576,99€/coche
Vagones tipo <<J>> propiedad de Renfe		
Varios de carga		0,1931€/Tm

Tabla 2. Tarifas aplicadas en el proceso de transporte. Elaboración propia con la información del convenio. [10]

3. Herramientas empleadas

Actualmente existen multitud de dispositivos móviles que se han convertido en un elemento fundamental en nuestro entorno, por características como su portabilidad, flexibilidad, que son fácilmente programables y con gran facilidad de manejo. De esta forma, a medida que adquieren mayor popularidad, los sistemas operativos con los que funcionan adquieren mayor importancia. Por ello, se ha realizado un estudio acerca de cuál es el sistema operativo más usado mundialmente y así determinar la plataforma para la que programar la app.

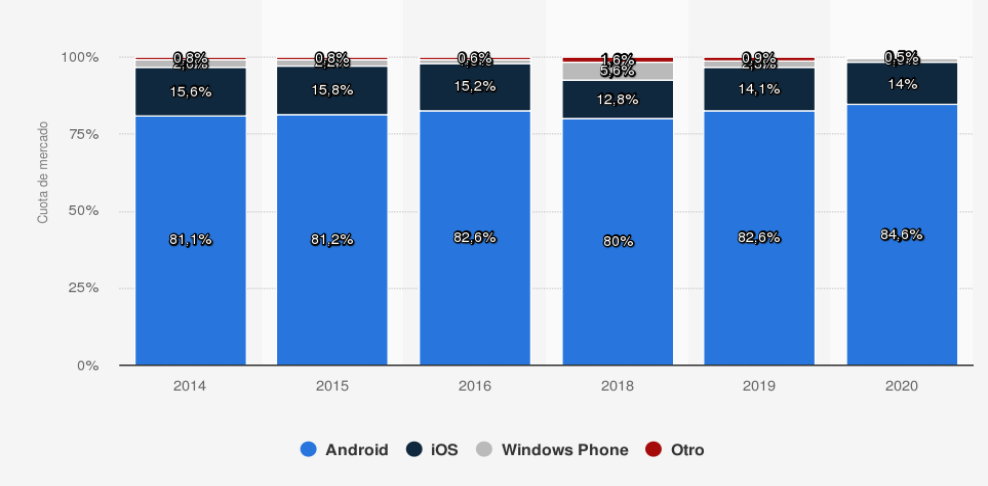


Ilustración 7. Cuota de mercado de pedidos de smartphones a nivel mundial entre 2014 y 2020. [13]

Como podemos comprobar en el siguiente gráfico, según los datos proporcionados por el IDC[5], los dispositivos con sistema operativo Android tiene la mayor cuota de mercado del mundo, alcanzando en los últimos 5 años valores próximos al 80% y con previsiones alcistas [13]. Además, el acceso al software y librerías de desarrollo es totalmente gratuito. Es por ello por lo que se ha decidido realizar la aplicación para la plataforma de Android, cuyas características serán explicadas a continuación.

3.1.Android.

Android es un sistema operativo basado en Linux con un entorno de ejecución basado en Java/kotlin que da soporte a gran variedad de dispositivos.

Este sistema, cuenta con un conjunto de herramientas y librerías propias (muchas de ellas compartidas con Java), así como infinidad de funciones integradas por el usuario al tratarse de una plataforma abierta. Además, es un sistema portable y adaptable a diferentes tipos de

[5] **International Data Corporation (IDC)** es el principal proveedor mundial de inteligencia de mercado, servicios de consultoría y eventos para los mercados de tecnología de la información, telecomunicaciones y tecnología de consumo.[23]

hardware, no estando enfocado únicamente a smartphones o tablets; sino también a otra serie de dispositivos como relojes inteligentes o televisores, entre otros.

3.2.Entorno: Android Studio.

El desarrollo de este proyecto se ha llevado a cabo con la plataforma de Android Studio, esta se trata de la herramienta oficial (y gratuita) de Google para desarrollar aplicaciones en teléfonos móviles Android.

Esta plataforma cuenta con numerosas ventajas a la hora de programar por particularidades como: ofrecer la corrección de errores en línea, proporcionar una rápida compilación, además de tener todas las herramientas necesarias para programar en cualquier tipo de plataforma y/o dispositivo Android, proporciona la posibilidad de ejecutar la aplicación mediante el uso de emuladores (creando un dispositivo virtual) o directamente conectando el móvil al puerto USB del dispositivo.

Ilustración 8. Logo tipo Android Studio.

3.2.1. Java

Java es uno de los lenguajes de programación más ampliamente establecidos. Se ha optado por el empleo de este puesto que es uno de los lenguajes oficiales para el desarrollo de aplicaciones móviles en Android, junto con Kotlin que, a pesar de que está adquiriendo una creciente importancia por su productividad, practicidad y eficacia, la comunidad y los recursos disponibles (librerías, foros de ayuda etc.) todavía no están tan maduros en comparación con Java.

Entre las características más destacables de Java encontramos que es un lenguaje orientado a objetos, es muy flexible; funciona en cualquier plataforma, dado que las aplicaciones que se desarrollan con este lenguaje funcionan en cualquier entorno; su uso no acarrea inversiones económicas, ya que no se precisa de ninguna licencia para programar; es de fuente abierta (u *open source*), lo que quiere decir que ofrece libre acceso a sus librerías nativas para sacar provecho de ellas y/o desarrollarlas; y es un lenguaje expansible, dado que es trabajado por una amplia comunidad de usuarios que ofrecen mejoras y soluciones constantemente. [14]

Ilustración 9. Logotipo Java.

3.3. Bases de datos

La base de datos se ha creado en un primer momento en un archivo Excel, siendo cada una de las hojas que lo componen una tabla de la base de datos. Posteriormente se guardaron con la extensión *.csv* (formato abierto sencillo con valores separados con comas) para ser importados, siendo previamente transformados al formato *.db* mediante el software *DB Browser for SQLite.*

SQLite es una biblioteca de C que implementa un motor de base de datos SQL de dominio público y, debido a que es de tamaño reducido es muy utilizable en los dispositivos móviles y viene por defecto integrada en el sistema Android.

Ilustración 10. Logotipo DB Browser for SQLite.

3.4.Código QR

El código QR (Quick Response o Respuesta Rápida) se trata de un código bidimensional que permite almacenar gran cantidad de información codificada de manera dinámica en la misma superficie que el código de barras tradicional y ofrece la ventaja de la lectura de este a alta velocidad y mayor capacidad de almacenaje de información. Además, tiene otras variantes como los códigos de corrección de errores[6] que ofrecen la capacidad de recuperar información perdida.

La ventaja de estos códigos es que están al alcance de cualquier usuario. Existen multitud de páginas web gratuitas con las que generarlos, donde solo es necesario asignarle la información que queremos que resulte en el mismo para posteriormente descargarlo. Para leerlos con un dispositivo móvil, normalmente se necesita la ayuda de alguna aplicación externa. En el caso de este proyecto, la aplicación resultante incorporará la capacidad de leer códigos QR a través de la librería *ZXing*.

Para el desarrollo de esta aplicación, por simplicidad, se ha optado por utilizar códigos QR estáticos asignados a cada material, de forma que la información queda directamente almacenada en los gráficos del código y, dichos datos asociados al mismo permanecen invariables.

[6] Los códigos de corrección de errores (también conocidos como **códigos Red Salomon** se aplican para restaurar los datos cuando falta parte del código QR o cuando este está dañado. Los niveles de corrección varían en función del área del código que está dañada.

3.4.1. Estructura de un código QR

Como ya se ha dicho, estos códigos son símbolos matriciales con una estructura en rejilla en forma de cuadrado [15]. Tiene varios patrones funcionales para que la lectura de los datos sea fácil y rápida. Los patrones que conforman el código son los que se muestran a continuación [16], [17]:

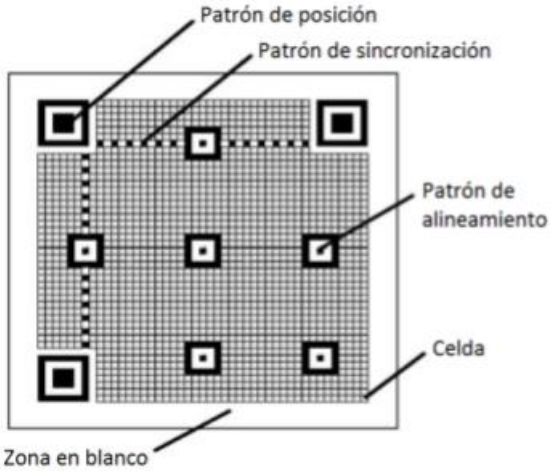


Ilustración 11.Estructura de un código QR [15]

1. **Celda o módulo**. Ese es elemento de menor tamaño del código QR. El código se compone en su totalidad por una combinación de estas de color blanco y negro.
2. **Patrón de posición**. Se trata de patrones ordenados en tres de las esquinas del código, lo que facilita la rápida detección del símbolo, el tamaño y la posición de este. Esto es, en mayor medida, lo que permite efectuar una lectura fácil y rápida del mismo, mejorando así la eficiencia de trabajo.
3. **Patrón de alineamiento.** Este sirve para corregir la distorsión y asegura que el código se pueda leer en el caso de que esté impreso sobre una superficie curva o distorsionada debido al ángulo. [18]
4. **Patrón de tiempo.** Se trata de un patrón de sincronización donde los módulos blancos y negros se ordenan de forma alternada para determinar la coordenada.
5. **Margen.** Se trata de la zona en blanco que se encuentra alrededor del código QR.
6. **Área de datos.** Es la zona donde los datos serán codificados, que es lo que representa la zona gris de la imagen. Esta información está cifrada mediante símbolos binarios, 1 y 0, que se convertirán en los módulos negros y blancos del cuadrado respectivamente. Además, esta área también incorporará los códigos de corrección de errores.

4. Desarrollo de la aplicación.

A continuación, se exponen las etapas que se han seguido durante el desarrollo de la aplicación:

1) **Conceptualización.** En esta etapa se parte de la idea propuesta en un principio, para que, a fin de cuentas, se adopte una formalización de esa idea. Todo ello se ha conllevado una minuciosa investigación acerca de todos los factores que afectan a este proyecto y los cuales se han redactado a lo largo de la presente memoria.
2) **Definición**. Con el objetivo de definir la funcionalidad del proyecto y asentar las bases de la aplicación, se determina el alcance del proyecto, cuál va a ser la complejidad del diseño, así como el lenguaje en el que se ejecutará la aplicación que, como ya se ha dicho anteriormente, será Java.
3) **Diseño**. Partiendo de la idea inicial de la aplicación, en esta fase se trata de materializar dicha idea, dándole un aspecto visual al objetivo que se quiere alcanzar, de tal forma que el usuario final pueda interactuar con ella de una forma fluida e intuitiva.
4) **Desarrollo.** Aquí se trata de "dar vida" a los diseños y crear la estructura sobre la que se va a apoyar el funcionamiento de la aplicación. Una vez que se decide emplear Java para implementar las funcionalidades, se procede a programar la parte lógica de la app.
5) **Depuración de errores.** Finalmente, una vez ejecutada y probada la aplicación, se harán las modificaciones pertinentes destinadas a mejorar la eficiencia de esta.

En base a esto, la aplicación se ha estructurado en base a la arquitectura por capas con el objetivo de organizar y optimizar la estructura del código fuente. Esta se compone de la capa de datos, capa de negocio y la capa de presentación; las cuales serán explicadas en los subsiguientes apartados. El código fuente de la aplicación será anexado al final de este documento como **ANEXO F**.

Por otro lado, el funcionamiento y las especificaciones técnicas vienen detalladas en el **ANEXO D**, donde se facilita el manual técnico (destinado al usuario) de la aplicación con sus pertinentes aclaraciones.

4.1. Capa de datos.

En primer lugar, esta es la capa que actúa como repositorio de datos. Las tablas de la base de datos, en una primera instancia, se crearon cada una en una hoja de cálculo de un libro Excel. Posteriormente se guardaron en formato ".csv" y se importaron en el programa DB Bowser para trabajar con estas en SQLite.

Para facilitar y automatizar la lectura de la base de datos, se ha creado una clase auxiliar, la cual se ha llamado *DBHelper*. Esta proporciona el canal de comunicación directo con la base de datos, heredando para ello la clase *SQLiteOpenHelper*, que es la que Android facilita para gestionar nuestra base de datos. [19]

La interacción que hace el sistema con esta capa sirve para realizar una serie de consultas a las diferentes tablas de la base datos (descritas a continuación) en función de las distintas necesidades de la aplicación. Por ejemplo, lecturas a la tabla de trayectos en función del seleccionado, dado que necesitamos saber la distancia de este para la posterior facturación, las consultas cada vez que se escanea un código QR, de plataforma y material respectivamente, para obtener los datos a mostrar en pantalla y los necesarios que se deben agregar al documento final a exportar.

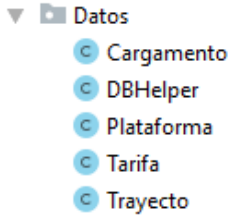


Ilustración 12. Clases de java de la capa de datos. Elaboración propia.

A continuación, se muestran las tablas que componen la base de datos con sus correspondientes campos.

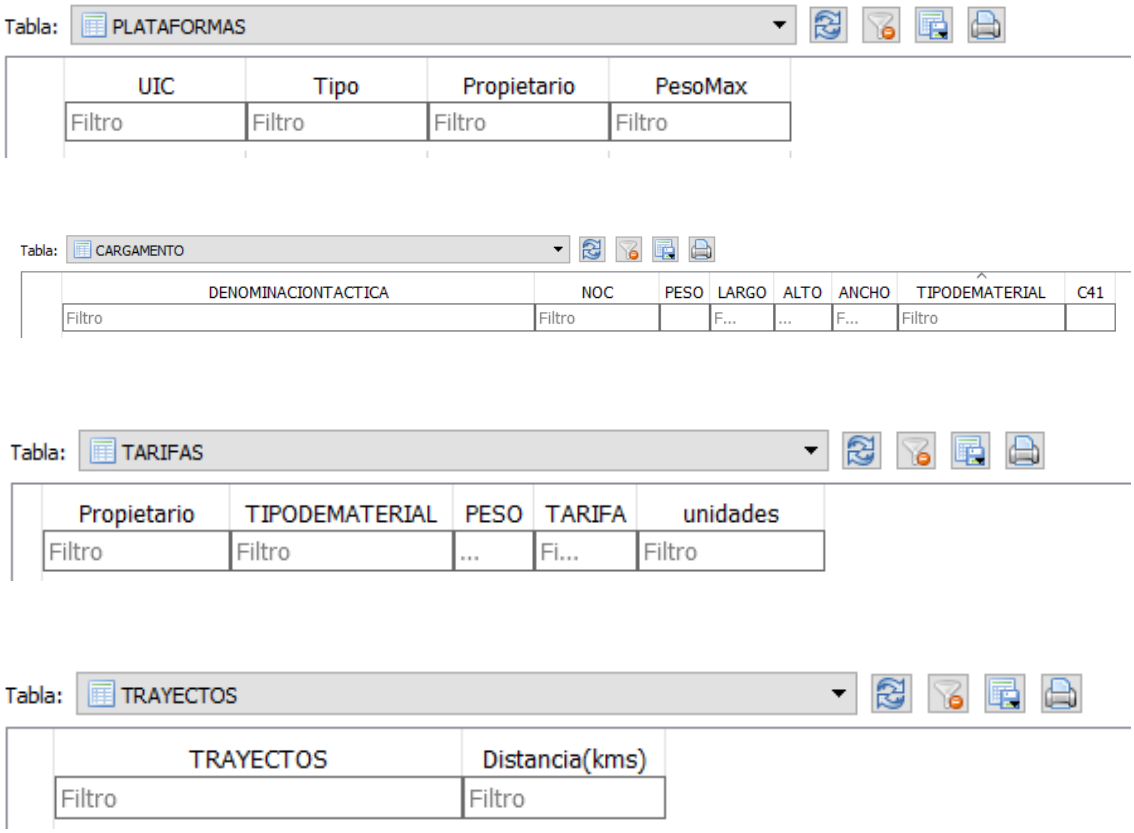


Ilustración 13. Encabezado de cada tabla de la base de datos (SQLite). Elaboración propia.

4.2. Capa de negocio.

Esta actúa como puente o intermediario entre la capa de presentación y la capa de datos. Es donde se programa la funcionalidad de la aplicación y donde se implementa la lógica de negocio, es decir, todo lo que la aplicación debe considerar antes de realizar una acción y el proceso que debe suceder a esta una vez que ha sido realizada. En nuestra aplicación corresponde con la estructura, donde se han programado las siguientes clases de Java con sus respectivas funciones:

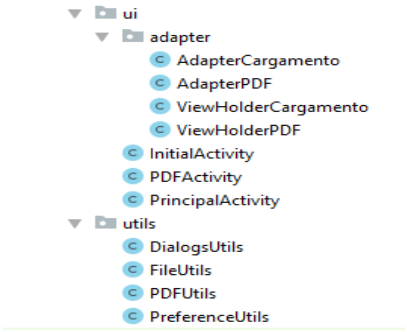


Ilustración 14. Clases de java de la capa de negocio.
Elaboración propia.

Sabiendo que estas son las clases lógicas que se implementan en la capa de negocio, se pasa a explicar brevemente las características principales de las mismas:

- **InitialActivity**. Esta es la pantalla de inicialización de la aplicación. Contiene únicamente un botón que envía al usuario a la pantalla principal, la cual permite ejecutar el resto de las acciones. Esta pantalla cumple su papel de presentación, con los colores e imagen corporativa, a la vez que evite que el usuario pase a la actividad principal por error (ej: apertura de una aplicación no deseada).
- **PDFActivity**. Esta es la función final que obtiene el listado de los ficheros de ayuda.
- **PrincipalActivity.** Aquí se ha desarrollado la parte lógica con más peso. Dentro de esta clase se han programado las principales funciones que debe proporcionar el sistema, como el escaneo de los códigos QR, las condiciones que deben cumplirse una vez efectuada la lectura, el acceso a los ficheros de ayuda en formato .pdf, la exportación del fichero a csv, así como todas las consideraciones que tiene en cuenta la aplicación para poder ejecutarse debidamente.

- **Subcarpeta adapter**:
    - **Adaptercargamento**. Se ha creado una clase para crear el listado de los cargamentos con un *Adapter*, la cual es una herramienta de Android que hace de "puente" entre los datos y un listado seleccionable de forma táctil por el usuario.
    - **Viewholdercargamento.** Esta herramienta sirve para gestionar y optimizar el uso de un *Adapter*, ya que el *ViewHolder* representa cada fila del listado (ej: darle formato). Gracias a esto se ha generado la lista expansible de los itinerarios disponibles.
    - **AdapterPDF.** Al igual que con los cargamentos, este recoge un listado de todos los documentos de consulta que proporciona la aplicación. En este caso, este

Adapter hace de puente entre los documentos almacenados en la carpeta *assets* y el listado de ficheros de ayuda que encontramos en el menú.

- **ViewHolderPDF.** Como en el caso de los cargamentos, el *ViewHolder* de los PDF representa cada elemento del listado de documentos, es decir, cada archivo PDF.

- **Paquete "utils".** Dentro de este paquete sean desarrollado clases las cuales son llamadas desde la *PrincipalActivity*. Entre las cuales se encuentran:
  - **DialogsUtils**. Esta clase se ha creado para crear de una forma más ágil los cuadros de diálogo emergente que aparecen en pantalla cada vez que se necesita la confirmación del usuario para realizar una acción.
  - **Fileutils**. Por último, esta clase es la que contiene la funcionalidad necesaria para generar el fichero de salida con la información almacenada gracias a la clase anterior.
  - **PDFUtils.** El objetivo de esta clase es ejecutar una tarea en segundo plano. De esta forma, la aplicación hace una copia temporal del fichero seleccionado para abrirse en una ruta concreta y Android se encarga de buscar las aplicaciones capaces de realizar dicha función, disminuyendo así la complejidad del código (ello implica que el dispositivo donde se instale tenga previamente instalado un lector de PDF).
  - **PreferenceUtils**. Esta es la clase donde se ha implementado como almacenar la información en el dispositivo de todos los registros que efectúa el usuario mediante la lectura de los códigos. De esta forma, es posible recuperar la información cuando el usuario minimiza la aplicación o apaga la pantalla del móvil.

4.2.1. Librerías usadas.

Las librerías (o *frameworks*) son grandes conjuntos de algoritmos, instrucciones y funciones preprogramadas que se pueden incorporar como un fragmento de código en el programa para agilizar tareas que ya han sido previamente desarrolladas, así como ciertas clases y funciones que facilitan el trabajo u objetivo que se desea alcanzar. Estas están destinadas a resolver un problema específico y tienen un propósito concreto, lo que reduce eficazmente la carga en el desarrollo. una de las ventajas de Java, y por tanto de Android, es la amplísima y activa comunidad de usuarios que brindan sus desarrollos de forma pública, proporcionando de forma libre una gran cantidad de herramientas para multitud de aplicaciones. A continuación, se exponen las librerías a las que se ha recurrido en este proyecto y la finalidad que se persigue con cada una de ellas:

- **ZXING.** La librería ZXing es un recurso que ofrece soporte para la lectura y decodificación para la gran mayoría de códigos de barras, códigos BIDI o QR en múltiples plataformas [20]. Como bien se ha dicho, esta ha sido integrada para poder realizar la lectura de códigos QR.
- **DEXTER.** Esta última es la encargada de gestionar todos los permisos que son necesarios confirmar por el usuario cuando se hace uso de alguna aplicación propia del dispositivo. En

este caso, ha administrado la gestión de los permisos destinados al uso de la cámara con sus respectivas preguntas.

4.3.Capa de presentación.

Esta última es la capa final donde se establece la interacción del usuario con el sistema, se trata de la interfaz de la aplicación, la cual se comunica únicamente con la capa de negocio y no contiene algoritmos sino archivos .xml que definen la apariencia de las diferentes pantallas u objetos visuales.

En cuanto al desarrollo de la parte gráfica de la aplicación, acorde con los requisitos que se han establecido, se diseñó previamente un boceto inicial, el cual ha resultado en la siguiente interfaz destinada al usuario. Esta se ha procurado que tenga el aspecto más "amigable" posible, es decir, que sea entendible y fácil de usar. Contiene todos los controles interactivos necesarios para poder cubrir los objetivos planteados y cuya función será explicada en el posteriormente.

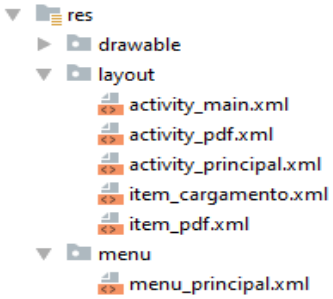


Ilustración 15. Diseño de las pantallas.
Elaboración propia.

Se trata de una serie pantallas o *Activities*: la primera (1) está enfocada a la presentación de la aplicación y la segunda, es la pantalla principal que ejecutará el resto de las funciones:

Ilustración 17. Pantalla de inicio de la aplicación. Elaboración propia.

Ilustración 16. Pantalla principal de la aplicación. Elaboración propia.

Conforme a la definición de los objetivos y acorde con estas pantallas, los controles interactivos o *layouts*[7] que encontramos son los siguientes:

- **Selector de trayectos (clase Android: Spinner)**. Lista desplegable con todos los itinerarios autorizados según el convenio MINISDEF-RENFE. La selección de este permitirá el posterior cálculo de la tasación del tren en función de la distancia correspondiente preestablecida en la base de datos.
- **Texto Índice de Plataforma (clase Android: Text View).** Muestra en número u índice de plataforma correspondiente del total en la que se encuentra el usuario en ese momento.
- **Flechas (clase Android: ImageButton)**. Los botones que se encuentran a los laterales de este texto permiten al usuario desplazarse de una plataforma a otra y ver su contenido y/o modificarlo.
- **Botón "Plus" (+) (clase Android: ImageButton)**. Este botón añade la "estructura" de una plataforma a la composición. En caso de no almacenar información en dicha estructura la aplicación proporciona la opción de desecharla a la hora de generar el cuadro de carga.
- **Botón "Delete" (papelera) (clase Android: ImageButton).** Al contrario que el caso anterior, este botón sirve para eliminar una plataforma. El hecho de eliminar la plataforma elimina a su vez el material que se haya almacenado en esta. Este botón al igual que los que se mencionan a continuación, las flechas, permanecen ocultos hasta que no se almacena información de, al menos, dos plataformas.
- **Botones seleccionables Plataforma/Material (clase Android: RadioButtons).** La elección entre el botón Plataforma o Material indica el tipo de lectura que va a realizar el escáner y en que tabla de la base de datos ha de buscar la información asociada al código escaneado.
- **Botón de escanear (clase Android: Button)**. Este redirige a la cámara para efectuar la lectura del código QR.
- **Texto de la información (clase Android: Text View)**:
    - **Escoja plataforma**. Este texto varía en función de la fase en la que se encuentre la aplicación. En primer lugar, aparecerá como una indicación al usuario: "escoja una plataforma". Una vez realizada la lectura este muestra el UIC de la plataforma y el tipo de esta.
    - **Cargamento.** Permanece oculto mientras no se haya realizado una lectura de tipo plataforma. Una vez realizada, aparecerá en este el número total de mercancías que ocupan la plataforma y la suma de sus pesos. El color de este texto varía en función de si dicha suma supera el máximo peso que es capaz de soportar la plataforma, en cuyo caso se mostrará en rojo[8].
- **Listado de material almacenado (clase de Android: RecycleView).** Lista donde se almacenan los materiales que ocupan una plataforma. En cada fila de esta se muestra

[7] Los *layouts* se consideran los contendores donde se organizan todos los controles interactivos.

[8] La representación gráfica de este "TextView" se puede apreciar en la ejemplificación del apartado 6 en la *Ilustración 19* con el ejemplo de una plataforma cargada. Del mismo modo, las flechas laterales y la papelera.

información sobre el material leído y un símbolo de cruz para, en caso de error, eliminarlo de la lista.

- **Botón generar cuadro de carga (clase Android: Button)**. Este es botón que genera en última instancia el fichero, con todos los datos almacenados, en formato csv.

Además, en el mencionado manual de usuario recogido en el **ANEXO D** se explica detalladamente, y de la forma más natural posible, paso a paso cómo funciona la aplicación y la secuencia que hay que seguir en el proceso de embarque del material en un transporte (con un ejemplo desarrollado completamente), así como todas las consideraciones se han tenido en cuenta a la hora de su desarrollo.

4.3.1. Funcionalidad de la aplicación

Para explicar la funcionalidad de la aplicación se ha usado una herramienta bastante gráfica y fácil de comprender que se usa comúnmente en los sistemas orientados a objetos. Se trata del modelo de casos de uso. Este es un tipo de notación UML (*Unified Modeling Language* en inglés) o "lenguaje unificado de modelado" que sirve para especificar y describir los métodos y/o procesos que permiten modelar el comportamiento de un sistema, concretamente identificando los requisitos funcionales del mismo.

En el siguiente diagrama se muestran los casos de uso asociados al actor dentro del sistema. Estos a su vez están relacionados mediante líneas de comunicación que definen la interacción usuario-sistema con dicha funcionalidad y además muestra la relación, en caso de que la haya, entre los propios casos de uso.

Una vez identificado cual será nuestro entorno del sistema, es decir, el software que vamos a representar, que en este e caso es la aplicación "*CARGA DE TRENES MILITARES*", identificamos al actor externo que interactúa con el sistema, siendo este el ROC y personal del CCMR, y los casos de uso, representados por un conjunto de funcionalidades que el sistema proporciona a los actores y a las que pueden acceder libremente desde este.

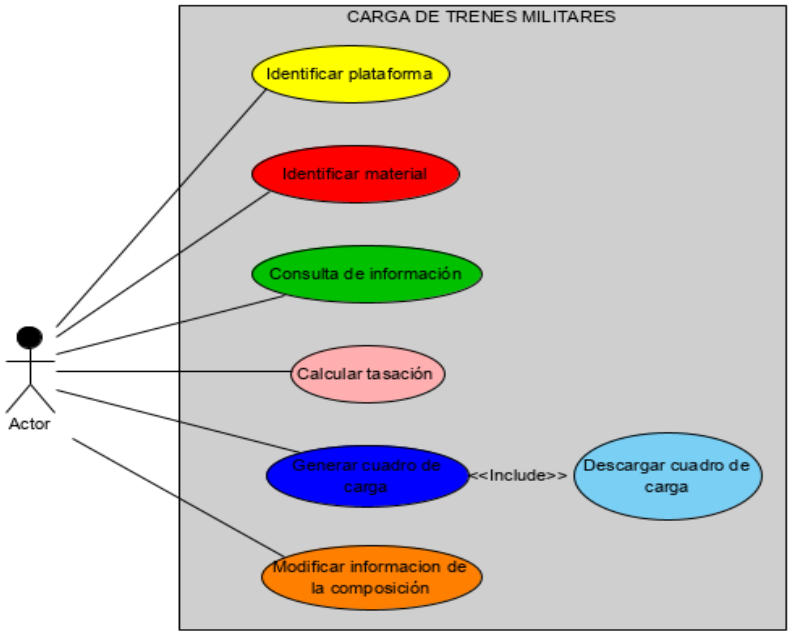


Ilustración 18. Diagrama de casos de uso. Elaboración propia.

- **Identificar plataforma**. El usuario puede obtener la información de una plataforma tras leer su respectivo código QR. Posteriormente, obtiene los siguientes datos de esta: código identificador UIC y tipo de plataforma (Obtenido a partir del identificador en la base de datos).
- **Identificar material**. Del mismo modo que el caso anterior, tras leer el código qr del material correspondiente, en pantalla se muestra de forma inmediata la denominación táctica del material y su peso (de nuevo obteniendo los datos de la base de datos a través del identificador escaneado en el QR).
- **Consultar información**. El usuario puede acceder a información a través del menú de la aplicación. De esta forma, puede consultar los planos de amarres y las fichas de gálibo de los materiales, las normativas que se aplican en el transporte por ferrocarril (convenio MINISDEF-RENFE y consigna C-41) y las prescripciones particulares de los materiales afectados por las citadas normativas.
- **Calcular tasación del tren**. Una vez que se genera el cuadro de carga, en el fichero descargado podemos obtener el valor de la facturación del tren.
- **Modificar información de la composición**. El usuario puede modificar la información de la composición, actualizando el listado de plataformas y los cargamentos que las ocupan.
- **Generar cuadro de carga.** Una vez almacenada la información necesaria de la composición del tren, el usuario puede generar el cuadro de carga, cuyo resultado será un fichero con extensión ".csv" en el que se muestran las plataformas del tren y el material embarcado en estas en el orden que el usuario las ha ido leyendo. Además, el fichero contiene información adicional de ambos elementos respectivamente.
    - **Descargar cuadro de carga**. Este caso de uso contiene una relación de inclusión con el caso de uso anterior, lo que quiere decir que siempre que se genere un cuadro de carga, este automáticamente procede a su descarga.

4.4. Análisis de resultados y depuración de errores.

En las pertinentes pruebas de la aplicación, hubo tropiezos con algunos detalles que no se habían tenido en cuenta y los cuales se fueron resolviendo a medida que surgían inconvenientes en su puesta en marcha. Entre ellos, se muestran a continuación algunos de los fallos encontrados y la solución implementada para su corrección:

⊗ La lectura de un código QR que no pertenezca a nuestra base de datos.
✓ Lo que provocaba el cierre inmediato de la aplicación ha pasado a convertirse en la emisión de un mensaje que comunica que el código escaneado no es válido y permite continuar con su uso adecuadamente.

⊗ Teniendo en cuenta el error humano se han tenido varias consideraciones, estas son las siguientes:
✓ No permite que la primera lectura sea de un material. Ha de ser de una plataforma para que ese material tenga donde almacenarse. La aplicación lanza un mensaje avisando al usuario: "Debe escoger una plataforma".

- ✓ Si el usuario pretende leer dos plataformas seguidas, avisa al usuario acerca del cambio de plataforma y solicita confirmación.
- ✓ La eliminación de un cargamento almacenado en la lista. Además, cuando la suma de los pesos de los materiales que ocupan una plataforma es mayor que el peso máximo que es capaz de soportar la plataforma, la aplicación lanza un mensaje avisando al usuario: "Peso máximo excedido", impidiendo a su vez que esta plataforma sea guardada. Debido a ello, se ha añadido a la lista de cargamentos un botón que permita el borrado del material que se crea conveniente para poder continuar con la lectura de códigos.
- ✓ Detecta si el usuario ha creado una plataforma y no tiene información sobre la misma, considerándola como "no válida". El sistema solicita confirmación para desecharla y no tenerla en cuenta a la hora de generar el fichero, eliminándola automáticamente.

- ⊗ Modificar la información de las plataformas y sus cargamentos.
- ✓ La posibilidad de navegar de una plataforma a otra. Esto permite que el contenido de estas sea modificado y actualizado previamente a generar el informe.

- ⊗ Perdida de información almacenada.
- ✓ Recuperar la información en caso de que el dispositivo se apague o el usuario salga de la aplicación temporalmente.

Para probarla finalmente, se ha tomado como ejemplo uno de los transportes ya realizados por la Cía. de Ferrocarriles, el cual servirá a su vez de ejemplo para explicar la secuencia a seguir por el usuario, y será expuesto en el apartado de la comprobación (apartado 6). El resultado que recoge el documento final es el siguiente:

1. **Trayecto**. Muestra el trayecto que realiza el transporte.
2. **Fecha**. Fecha en la que se efectúa el ejercicio.
3. **Campos de la composición**. El contenido que se muestra en la tabla sigue este orden: plataforma; tipo de plataforma; denominación táctica; peso; dimensiones, C-41 y tipo de material.
4. **Tabla resumen**. Al final del documento aparecen dos tablas a modo resumen de la composición del tren. En primer lugar, se muestran las plataformas agrupadas por tipo y la cantidad de cada uno de ellos. Por consiguiente, se muestran los materiales agrupados por nombre, la cantidad de cada uno y el sumatorio total de sus pesos.
5. **Facturación**. Junto a las tablas anteriores aparece el valor final de la tasación del tren (sin IVA incluido).

6. Comprobación.

Para la puesta en escena de la app de *CARGA DE TRENES MILITARES*, se aprovechó que la Cía. de Ferrocarriles del Regimiento de Pontoneros y Especialidades de Ingenieros nº12 (MING) participó durante los días 9 y 10 de octubre en el transporte por ferrocarril de vehículos acorazados y mecanizados y personal, de la Brigada "Guzmán el Bueno" X, desde Córdoba hasta el CENAD San Gregorio (Zaragoza), para la realización del ejercicio BETA "CERVANTES 19".

El proceso de transporte entre la estación de mercancías del Higuerón y la estación militar de ferrocarril de San Gregorio fue gestionado por el CCMR y la Cía. de Ferrocarriles del BESPII/12, en coordinación con RENFE, constituyendo dos Terminales Terrestres Ferroviarias.

Teniendo este transporte como ejemplo base para validar la aplicación con sus respectivos destinatarios, se imprimieron los códigos QR asociados a los materiales y plataformas que conformaban el ejercicio, formando así la composición de los trenes sobre papel y simulando la integración de la aplicación en el proceso de embarque del material. La composición de las plataformas con su respectivo material embarcado eran los siguientes.

NUMERACIÓN (UIC)	CONTENIDO
507105080059	Z118006
837139710029	CC LEOPARDO 2E
837139710201	CC LEOPARDO 2E
837139710433	CC LEOPARDO 2E
837139710011	CC LEOPARDO 2E
837139710144	CC LEOPARDO 2E
837139710060	CC LEOPARDO 2E
837139710276	CC LEOPARDO 2E
837139710177	CC LEOPARDO 2E
837139710219	VEC M1
837139710136	VEC M1
837139710128	CREC LEOPARDO 2ER "BÚFALO"

NUMERACIÓN (UIC)	CONTENIDO
507105080034	Z118004
837139710037	CC LEOPARDO 2E
837139710052	CC LEOPARDO 2E
837139710102	CC LEOPARDO 2E
837139710086	CC LEOPARDO 2E
837139710383	CC LEOPARDO 2E
837139710466	CC LEOPARDO 2E
837139710532	VCI/C PIZARRO FASE II
837139710540	VCI/C PIZARRO FASE II
837139710375	VCI/C PIZARRO FASE II
837139710482	VCI/C PIZARRO PC BON
837139710516	VCI/C PIZARRO PC BON
837139710284	2 TOA M113 MCCLA TOW
837139710169	CC M47 ER3

Tabla 3. Composición de los trenes. Elaboración propia

A continuación se muestra la representación gráfica de la composición simulada del segundo tren (tabla derecha) para poder hacer su respectiva comprobación junto con sus códigos QR asociados: los que se encuentran por encima de los vehículos corresponden a los materiales; los que están por debajo de la vía, del mismo modo, corresponden a las plataformas. Además, junto con esta última, se presenta la captura de la pantalla de una de las plataformas con su cargamento correspondiente de manera demostrativa:

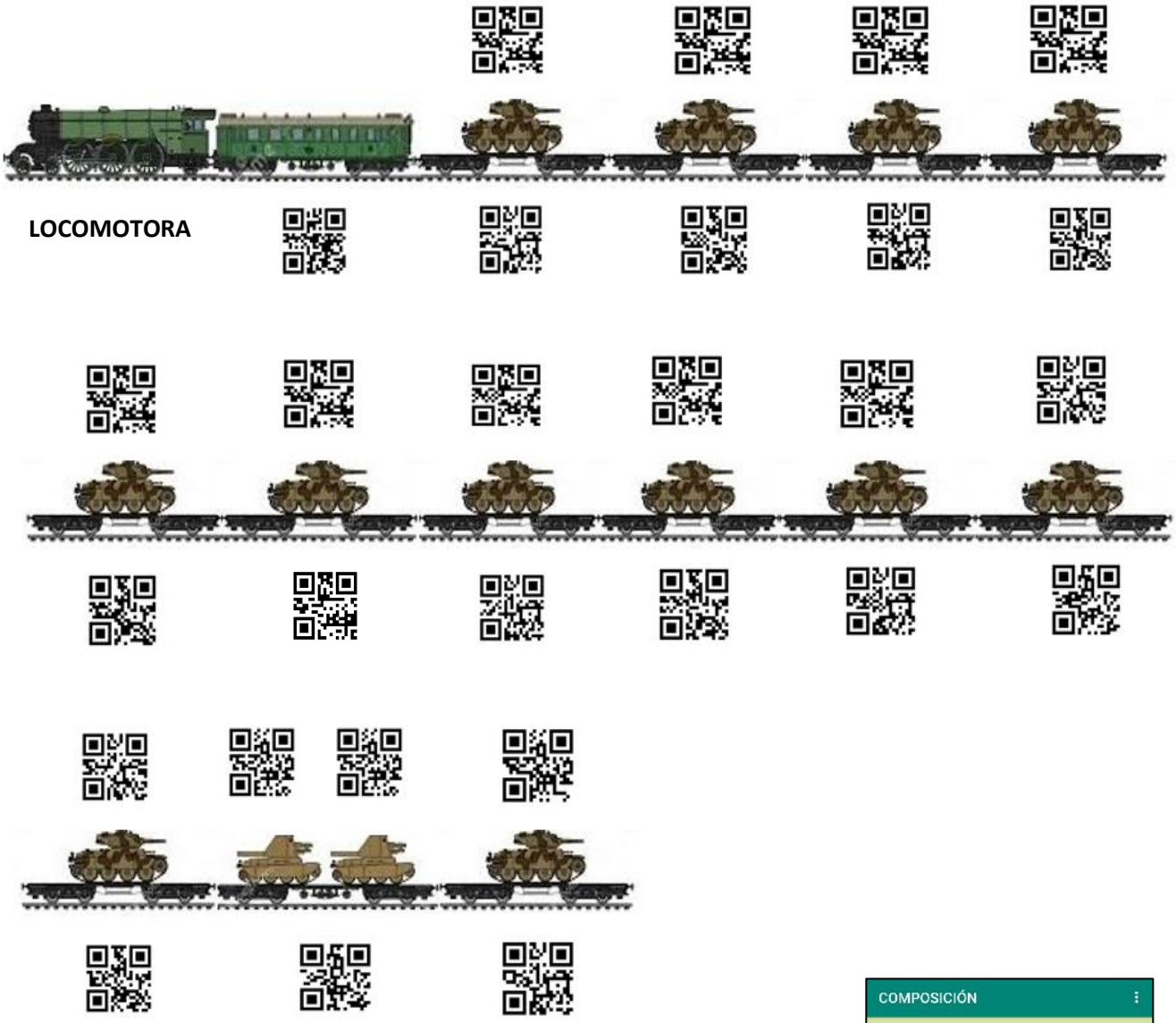


Ilustración 19. Representación grafica de la composición con sus respectivos QR asociados. Elaboración propia.

Como podemos comprobar en este ejemplo, se trata de una captura de pantalla de una de las plataformas del tren leídas con su cargamento correspondiente. En este caso se trata de la plataforma 13, con código UIC-837139710284 (PMMER) y cuyo cargamento son dos vehículos TOA (Transporte Oruga Acorazado). Podemos ver como la aplicación muestra en pantalla estos datos tras haber leído sus códigos correspondientes, además del resto de elementos que aparecen en la pantalla actual ya mencionados en el apartado anterior.

Ilustración 20. Ejemplo pantalla de la plataforma 13 cargada. Elaboración propia.

En el **ANEXO E** podemos ver como los documentos oficiales tramitados en el transporte y, a continuación de estos, en el mismo anexo, se adjuntará el resultado generado por la aplicación tras haber realizado la lectura de los trenes. De esta forma, se podrá hacer una clara comparación con los resultados obtenidos.

TRAYECTO: Córdoba-El Higuerón -San Gregorio

28/10/2019

COMPOSICIÓN

UIC	TIPO	DENOMINACIÓN TÁCTICA	PESO	C-41	ALTO	LARGO	ANCHO	TIPO DE MATERIAL
507105080034	BC10X							
837139710037	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710052	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710102	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710086	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710383	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710466	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710532	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710540	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710375	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710482	PMMER	VCI/C PIZARRO PC BON	24	null	2932.0	6836.0	3.15	Vehículos blindados
837139710516	PMMER	VCI/C PIZARRO PC BON	24	null	2932.0	6836.0	3.15	Vehículos blindados
837139710284	PMMER	TOA M-113 A1 MCC TOW	10	null	2.45	4.78	2.64	Vehículos blindados
837139710284	PMMER	TOA M-113 A1 MCC TOW	10	null	2.45	4.78	2.64	Vehículos blindados
837139710169	PMMER	CC M-47 ER3	45	X	3.07	8.4	3.62	Carro de combate

TIPO PLATAFORMA	NÚMERO
PMMER	13
BC10X	1
TOTALES	14

DENOMINACIÓN TÁCTICA	NÚMERO	PESO
VCI/C PIZARRO FASE II	3	28.0
VCI/C PIZARRO PC BON	2	24.0
TOA M-113 A1 MCC TOW	2	10.0
CC LEOPARDO 2 E	6	63.0
CC M-47 ER3	1	45.0
TOTALES	14	575.0

COSTE TOTAL (IVA no incluido): 71.378,38 €

Ilustración 21. Resultado que proporciona la aplicación de la composición del tren probado. Elaboración propia.

El resultado obtenido resultó totalmente satisfactorio, teniendo en cuenta que, en base a los objetivos planteados, se han cumplido todos:

- ✓ Reducción eficazmente del tiempo empleado en el proceso de carga. La lectura de los códigos conllevó un tiempo de 8 minutos, comparado con los 90, aproximadamente, que hubiera costado hacerlo de forma tradicional. Sin embargo, ha de tenerse en cuenta que la comprobación se puso sobre papel, a lo que habría que sumarle el desplazamiento entre las distintas plataformas, y el uso eficiente de la aplicación por parte del ROC.
- ✓ Las fichas de galibo resultan de gran apoyo para realizar rápidas consultas y resolver pequeñas incidencias que surgen durante el proceso de embarque, principalmente para centrar los vehículos adecuadamente en las plataformas.
- ✓ El reconocimiento inmediato del material cargado gracias a la lectura de código QR facilita la automatización del proceso, evitando así equivocaciones humanas que puedan generar agravantes en el proceso.
- ✓ Como se ha comprobado, la aplicación es capaz de realizar la tasación del tren y, gracias a proporcionar estos datos de manera inmediata al CCMR facilita la comprobación y el control que se ha de gestionar por parte de este.

7. Conclusiones y líneas futuras.

7.1. Objetivos alcanzados.

En primer lugar, el mayor inconveniente que se ha presentado a la hora de desarrollar este proyecto ha sido la falta de experiencia en el ámbito de la programación, y partir el aprendizaje desde cero teniendo que aprender el qué era y cómo había que desarrollar cada uno de los componentes que conforman la aplicación final.

A pesar de ello, como ya se ha mencionado en el apartado anterior, el proyecto ha sido capaz de paliar todos los objetivos abordados: agiliza la tarea que desempeña el ROC a gran escala; facilita la información necesaria en caso de necesidad de consulta de esta; calcula la tasación del tren y permite el reconocimiento inmediato de los elementos de una composición

Además, la aplicación actual cuenta con varios aspectos con los que no se había contado al principio, como por ejemplo la autorrecuperación de la información registrada en caso de que la aplicación se pare o el dispositivo se apague o la opción de modificar información en caso de equivocación. Por ello, se ha ido adaptando el diseño del interfaz a medida que se ha ido desarrollando el proyecto, haciendo las variaciones necesarias en el código para que al final esta presentase de la forma más intuitiva posible, facilitando del mismo modo su manejo.

En conclusión, de este proyecto se han obtenido unos conocimientos y habilidades muy positivas a pesar de trabajar con tecnologías con las que no se había tratado anteriormente. Se ha generado una aplicación funcional que ha cumplido los objetivos que se habían planteado en un primer momento. Además, la Cía. de Ferrocarriles y el personal del CCMR, ya están trabajando la posibilidad de implementar oficialmente la aplicación en el proceso de carga de los trenes.

7.2. Líneas futuras.

Durante la realización de este trabajo también se han planteado otras variantes, las cuales dejan la cabida a un margen de ampliación.

Por un lado, debido al incremento de esa amenaza en el flanco este de Europa y al fomento del transporte por ferrocarril mencionada al inicio de esta memoria, queda la posibilidad de que esta aplicación abarque el marco internacional, generando los códigos QR cuyo identificativo sea el código OTAN que esté asignado a ese material. Además, en un futuro también sería posible implantar dichos QR de carácter dinámico, de forma que la información contenida no contenga directamente la identificación del material, sino que redirija a una *url* cuyo contenido sea controlable de forma externa.

Por otra parte, estudiar la viabilidad de mostrar el proceso de embarque de material en línea. De esta forma, permite que el órgano encargado de gestionar esta información, el CCMR, tenga la posibilidad de estar al corriente en todo momento del desarrollo del proceso de carga.

Además, se puede implementar a la aplicación una consulta particular de la información de un material cuando se efectúa una lectura de este. De esta forma, cada vez que se escanee un

código se pueda acceder directamente a las fichas de galibo y documentación adicional perteneciente al mismo en vez de buscar el documento en el menú actual de la aplicación.

Por último, otra de las variables necesarias para la ejecución real de este proyecto es el estudio de los posibles materiales con los que grabar el código QR (tipo pintura o pegatina) e incorporar al cargamento, que presenten la aptitud para cualquier tipo de condiciones meteorológicas adversas a las que puede estar sometido un tren en un transporte.

Bibliografía

[1] Unkonwn, “Países del flanco oriental de la OTAN piden una Alianza más fuerte,” 2018. [Online]. Available: https://www.lavanguardia.com/politica/20180608/444216732059/paises-del-flanco-oriental-de-la-otan-piden-una-alianza-mas-fuerte.html. [Accessed: 10-Sep-2019].

[2] J. R. Arévalo, “LLegar. Manual Técnico de operaciones.” (PE-07), 2007.

[3] “NAI2403/17. Seguimiento y servicios de Ferrrocaril.” Mando de Ingenieros (RPEI12-BESPII/12).

[4] “SUBSISTEMA DE TRANSPORTE.” Norma General 02/13. Estado Mayor del Ejército de Tierra, 2013.

[5] “Instrucción Técnica 09/02. Procedimiento operativo sobre transporte en trenes miliares especiales.” Mando de Apoyo Logístico del Ejército (Dirección de Transportes), p. 15, 2002.

[6] “NAI 2403/09. SOBRE SEGUIMIENTO Y CONTROL DE TRENES MILITARES.” Plana Mayor de Mando (Regimiento de Pontoneros y Especialidades nº12), 2009.

[7] “PROCEDIMIENTO PARA LA PREPARACIÓN Y EJECUCIÓN DE LOS TRANSPORTES INCLUIDOS EN EL PROGRAMA ANUAL DE TRANSPORTE TERRESTRE (PAT-T).” Apéndice 1 al ANEXO VII (PAT-T) del PROAL DINFULOG 2019. DIRECCIÓN DE INTEGRACIÓN DE FUNCIONES LOGÍSTICAS . SUBDIRECCIÓN DE GESTIÓN., 2019.

[8] “Normas para la gestión de Peticiones de transporte, Órdenes de transporte y utilización del impreso T-5OO.” ESTADO MAYOR DEL EJÉRCITO., 2006.

[9] ADIF, “Consigna serie C-41. Normas para el transporte de vehículos militares de caracaterísticas excepcionales.” Dirección de Seguridad en la Circulación, p. 46, 2012.

[10] “Manual técnico. Transporte por ferrocarril.” (MT5_007) Mando de Adiestramiento y Doctrina.Dirección de Doctrina, Orgánica y Materiales., p. 130, 2000.

[11] BOD, “Convenio entre el Ministerio de Defensa y Renfe-Operadora para el transporte de mercancías y personal.” Boletín Oficial del Ministerio de Defensa., p. 36, 2018.

[12] Renfe Mercancías, “Normativa de cargamento. NC-007-07-17.” Dirección General de Mecancías. Gerencia de Seguridad y Autoprotección., 2017.

[13] Statista Research Department, “Cuota de mercado de sistemas operativos para smartphones por pedidos 2014-2020.,” 2019. [Online]. Available: https://es.statista.com/estadisticas/600731/cuota-de-mercado-de-sistemas-operativos-para-smartphones-por-pedidos--2020/. [Accessed: 05-Oct-2019].

[14] J. Pérez and A. Gardey, “Definición de Java,” 2013. [Online]. Available: https://definicion.de/java/. [Accessed: 10-Oct-2019].

[15] M. Lozano, “DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA CONTROL REMOTO DE UN SERVICIO WEB.” UNIVERSIDAD CARLOS III DE MADRID. ESCUELA POLITÉCNICA SUPERIOR., 2012.

[16] Unkonwn, “¿Qué es el código QR?,” *KEYENCE*. [Online]. Available:

https://www.keyence.com.mx/ss/products/auto_id/barcode_lecture/basic_2d/qr/index.jsp.

[17] J. F. Avila de Tomás, "¿Qué es un código QR?. Definición y estructura," *Grupo de Nuevas Tecnologías de la SoMaMFyC*, 2012. [Online]. Available: https://nuevastecsomamfyc.wordpress.com/2012/05/08/que-es-un-codigo-qr-definicion-y-estructura/.

[18] Unkonwn, "Funcionamiento de un código QR," *uQR.me*, 2016. [Online]. Available: https://uqr.me/es/qr-code-generator-marketing/como-funciona-codigo-qr/. [Accessed: 20-Oct-2019].

[19] IAvilaE, "Programación Android, Base de Datos II," 2013. [Online]. Available: http://www.proyectosimio.com/es/programacion-android-base-de-datos-ii/.

[20] Unkonwn, "Librería zXing para lectura de códigos QR en Android," *BILJET APP*, 2013. [Online]. Available: https://biljetapp.wordpress.com/2013/03/04/presentacion/. [Accessed: 12-Oct-2019].

[21] "Inspección General nº.66. Sistema de Apoyo Logístico." Estado Mayor del Ejército, 2012.

[22] Renfe, "Catálogo de vagones." 28036 Madrid, 2015.

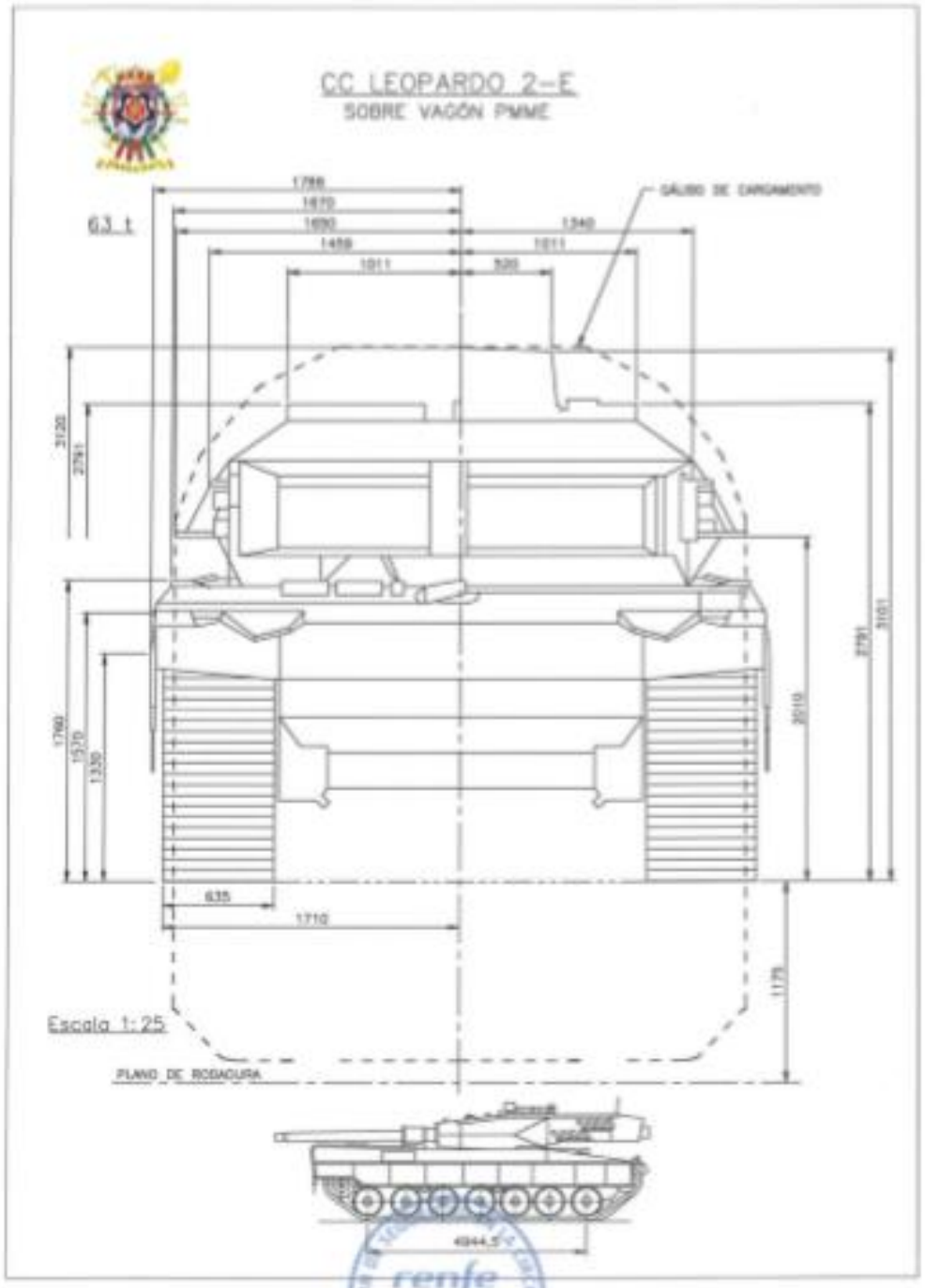
[23] IDC, "Sobre IDC," *Información por sector dirigida a compradores de tecnología*, 2019. [Online]. Available: https://idcspain.com/sobre-idc. [Accessed: 05-Oct-2019].

ANEXOS

ANEXO A. FICHA DE GÁLIBO.

Ejército de Tierra

USO OFICIAL



renfe Seguridad Normativa

USO OFICIAL

ANEXOS

ANEXO B. PLATAFORMAS: DETALLES TÉCNICO.

TIPOS DE VAGONES

CARACTERÍSTICAS

MODELO	COCHE DE VIAJEROS				
Denominación	Serie 10.000		Serie 12.000		Literas 9.600, 10.600, 12.600
Tipo	A10X 10.000	A12X 10.200	A10X 12.000	B12X 12.300	X
Clase	1.ª	2.ª	1.ª	2.ª	X
Departamentos	10	11	10	12	10
Plazas	60	88	60	96	posición día: 80 posición noche: 60
Figura	C.1	C.2	C.3	C.4	C.5

MODELO		PLATAFORMA DE CARGA						
Denominación		M1	M2	MM2 (año 66)	MM2 (año 73)	MQ	MMQ	PMM-E
Carga máxima (t)		27,5-27,0	20	50,0-55,0	55,2-56,0	25,9	55,3	57,5
Altura piso (m)		1,23	1,25	1,33	1,28	1,23	1,28	1,175
Dimensiones Interiores	Largo (m)	12,5	8,62-7,93	18,45	18,5	12,5	18,5	11,26
	Ancho (m)	2,77	2,84	2,75	2,74	2,77	2,74	3,29
	Altura costado (m)	0,45	0,42	X	X	X	X	X
	Altura telero/piso (m)	1,18-2,01	X	1,21	1,19	1,18	1,19	X
	Altura testero (m)	X	X	X	X	X	0,4	0,42
	Superficie útil (m^2)	34,6-35,0	24,5-22,5	50,55	50,7	34,6	50,7	37
Costado		Abatibles	Abatibles	X	X	Abatibles	X	X
Testeros		Abatibles	Abatibles	Abatibles	Abatibles	Abatibles	Abatibles	Abatibles
Teleros		Abatibles	X	Abatibles	Abatibles	Abatibles	Abatibles	X
Figura		C.6	C.7	C.8	C.9	C.10	C.11	C.12

MODELO		VAGONES CERRADOS DE CARGA		
Denominación		JPD-1	J-1	J-2
Carga máxima (t)		23,9	25,2	28,0-27,5
Altura piso (m)		1,25	1,25	1,23
Dimensiones Interiores	Largo (m)	12,73	12,73	9,29
	Ancho (m)	2,64	2,64	2,70-2,72
	Alto (m)	2,84	2,84	2,77-2,80
	Superficie útil (m^2)	33	33	25,08-25,27
	Volumen útil (m^3)	72	80	50,16-50,52
	Ancho útil (m)	5,92	2,5	2
	Alto útil puerta (m)	2,3	2,15	2
Figura		C.13	C.14	C.15

ANEXOS

ANEXO C. MODELOS DE DOCUMENTACIÓN EN EL PROCESO DE TRANSPORTE POR FERROCARRIL.

Apéndice 1. Programa Anual de Transporte Terrestre (PAP-T).

Resumen transporte BRIX.

	DATOS IDENTIFICACIÓN						MEDIOS Y/O PASAJEROS A TRANSPORTAR													MOVIMIENTOS PEVISTOS									
							VEHICULOS														IDA					REGRESO				
								MILITAR								CIVIL													
								AGTP		GTP III/21		GTP III/41		GL III/61		CIVIL CTRA		CIVIL FFCC										
	MANDO	Nº SERIE	Nº SERIE PAT-M	PRIORIDAD	NOMBRE UNIDAD	MOD. VEH.	Nº VEH.	I	V	I	V	I	V	I	V	I	V	I	V	Nº PAX	ORIGEN	FECHA CARGA	FECHA SALIDA	FECHA DECARGA	DESTINO	ORIGEN	FECHA CARGA	FECHA SALIDA	FECHA DECARGA /	DESTINO
BRI X																				500	BASE CERRO MURIANO	11-oct	11-oct	11-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
	BRI X	96		2	BRI X (VJTF 18)	CARRO COMBATE LEOPARDO 2 E	14											14	14		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						CREC LEO 2ER	1											1	1		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						VCI PIZARRO PC	2											2	2		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						VCI PIZARRO FASE II	28											3	3		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						CZ	1											0	0		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						TOA MP 120	8											0	0		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						TOA M-548 CARGA	9											0	0		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						TOA M-113	10											2	2		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						TOA M-577	1											0	0		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						TOA MCCLA	10											0	0		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						TOA VCZ	6											0	0		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						VLP	1											0	0		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						CREC M-47	1											1	1		BASE CERRO MURIANO	10-oct	10-oct	10-oct	CENAD S. GREGORIO	CENAD S. GREGORIO	22-oct	22-oct	22-oct	BASE CERRO MURIANO
						TOTAL	92	0	0	0	0	0	0	0	0	0	0	23	23	500										

ANEXOS

<u>Apéndice 2</u>. Cuadro de composición para un programa de transporte.
(Propuesta de Organización de Trenes)

FUERZA TERRESTRE — **MANDO DE INGENIEROS**
REGIMIENTO DE PONTONEROS Nº 12 — **BESP II/12**
CELULA CONTROL MOVIMIENTO POR FERROCARRIL DEL ET

CUADRO DE COMPOSICION PARA UN PROGRAMA DE TRANSPORTE

PETICIONES DE TRANSPORTE Nº 50020648-19-530135 DE FECHA 3 de sep de 19
PUNTO 4.2. Apéndice 1 ANEXO(PAT_T) PROAL DINFULOG 201 DE FECHA 8 de dic de 18

TREN Nº TREN MO15-I/MO16-R

UNIDAD: BRI X(serie96) **FECHAS: IDA:** 09/10/ 2019 **REGRESO:** 21/10 /2019
ORIGEN: CORDOBA EL HIGUERON MERC (COR) **DESTINO:** SAN GREGORIO (ZAR)

EFECTIVOS				MATERIAL FERROVIARIO						
PERSONAL	OFICIALES SUPERIORES Y OFICIALES SUBOFICIALES TROPA TOTAL 50			Nº	TIPO	EJES	LT	TARA	TMM	TMB
				1	LITERA	4	26.40	50	2	52
MATERIAL MILITAR	8	CC LEOPARDO 2E		8	PMM-ER	32	100.00	170.4	504	674.4
	1	CREC LEOPARDO 2ER "BUFALO"		1	PMM-ER	4	12.50	21.3	63	84.3
	2	VEC M1		2	PMM-E	8	25.00	42.6	32	74.6
	11									
TOTALES				12	1	48	163.90	284.3	601	886.3

C-41: VEHÍCULOS QUE VIAJAN EN AMPARO DE LA CONSIGNA C-41

ZARAGOZA, 05 DE SEPTIEMBRE DE 2019
LA TTE 2ª JEFE CCMR

MARIA LUISA MAYO LOPE

Resumen Material Ferroviario

LITERA	1
PRIMERA	0
MMQC	0
PMM-E	2
M1	0
MM2	0
MA1	0
PMM-ER	9

(*) En cumplimiento de la IT 09/02 MALE, se utilizarán normalmente coches de primera clase; si el trayecto incluye seis o más horas dentro del tramo comprendido entre las 20'00 y las 07'00 del día siguiente se emplearán coches litera.

<u>IDA:</u>	SE SOLICITA LA <u>LLEGADA</u> A LA ESTON MILIT SAN GREGORIO A LAS 12:00 HORAS DEL DÍA 10 DE OCTUBRE DE 2019 SE SOLICITA EL INICIO DEL EMBARQUE A PARTIR DE LAS 09:00 HORAS DEL DÍA 08 DE OCTUBRE DE 2019
<u>REGRESO:</u>	SE SOLICITA LA LLEGADA A LA ESTACION CORDOBA EL HIGUERON MERCANCIAS A LAS 08:00 HORAS DEL DÍA 22 DE OCTUBRE DE 2019 SE SOLICITA EL INICIO DEL EMBARQUE EN LA ESTON. MILIT S. GREGORIO A PARTIR DE 08:00 HORAS DEL DIA 21 DE OCTUBRE DE 2019

ANEXOS

Apéndice 3. Justificación de petición de transporte. (T-500)

PT

1 **DOCUMENTO QUE JUSTIFICA LA PT**
ROPP - PAT-T 96/25 VEH. AC-MZ/CORDOBA-ZARAGOZA/I-R/FERROCARRIL

2 **FECHA** 3 Sep 19

3 **Nº DE REGISTRO** 50020648-19-530135

4 **PETICIONARIO, DIRECCIÓN** 50020648
CG BRI X - CERRO MURIANO (CÓRDOBA)
CTRA.CORDOBA-BADAJOZ, KM.253
Tf
Fx
Tx

5 **DESTINATARIO PT, DIRECCIÓN** 57220450
MALE_DIRECCION. - MADRID
CALLE DE PRIM Nº 6 Y 8

6 **REMITENTE DEL RECURSO, DIRECCIÓN** 50020648
CG BRI X - CERRO MURIANO (CÓRDOBA)
CTRA.CORDOBA-BADAJOZ, KM.253
POC BG. PULIDO (TLF. 699445907)
Tf
Fx
Tx

7 **SITUACIÓN DEL RECURSO, DIRECCIÓN**
EST. FF.CC. - EL HIGUERON (CÓRDOBA) C.P.14193
EST. FF.CC. MERCANCIAS
POC BG. PULIDO (TLF. 699445907)

8 **CONSIGNATARIO, DIRECCIÓN**
POC
Tf
Fx
Tx

9 **DESTINO FINAL DEL RECURSO, DIRECCIÓN**
EST. FF.CC. - EL HIGUERON (CÓRDOBA) C.P.14193
EST. FF.CC. MERCANCIAS
POC BG. PULIDO (TLF. 699445907)

10 **AMPLIACIÓN DE DATOS**
PLAZO DE EJECUCIÓN El 9 Oct 19
FECHA REGRESO El 21 Oct 19
OTROS DATOS
EX. Nº 96 DEL PAT-T 2019 / TRANSPORTE POR FERROCARRIL PARA 25 VEHÍCULOS AC-MZ / SALIDA: 09OCT19 EST. FF. CC. MERCANCÍAS DE CORDOBA (EL HIGUERÓN-CORDOBA) HASTA EL CENAD SAN GREGORIO (ZARAGOZA) - REGRESO: 21OCT19 RECORRIDO INVERSO / SE SOLICITA QUE CADA TREN DISPONGA DE UN (1) VAGÓN DE PASAJEROS PARA LA ESCOLTA T-S / POC BG. PULIDO (TLF. 699445907)

CÓDIGO CONTROL COSTES 19PREP9NTRANS
GASTO PREVISTO 248.791,00
EJ. PRESUPUESTARIO 2019 **Nº PREVISIÓN** 19-3826

PT RELACIONADA CON ESPECIAL (SOBREDIMENSIONADO)
FINALIDAD MILITAR USO EXCLUSIVO MILITAR
MEDIOS ET NO **A REALIZAR CON MEDIOS DEL UCO**
TIPO ESCOLTA S/T **UCO ESCOLTA** 50002280 - BIMZ II/2
VALOR TOTAL DE LA CARGA 99.999.999,99
COORDINAR CON OTRAS PTs NO

DESCRIPCIÓN DEL RECURSO

11 **PERSONAL**
OFICIALES
SUBOFICIALES
TROPA
TOTAL

ARMAMENTO INDIVIDUAL E IMPEDIMENTA

OTROS 12 **CLASE** 13 **VOLUMEN (M3)** 2.176,820 14 **PESO (KG)** 1.179.980,0

Nº ELEMENTOS	LARGO (M)	ANCHO (M)	ALTO (M)	RECURSO		CLASE
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,67	3,75	3,00	CARRO COMBATE LEOPARDO 2 E	BICC I/10	0706
1	9,07	3,54	2,74	CARRO REC LEOPARDO 2 ER	BICC I/10	0706
1	8,40	3,62	3,07	CARRO REC M-47 ER3	BIMZ II/2	0706
1	4,78	2,69	2,23	TOA M-113 A2 LZN MCCLA TOW-LWL	BIMZ II/2	0706
1	4,78	2,69	2,23	TOA M-113 A2 LZN MCCLA TOW-LWL	BIMZ II/2	0706
1	7,13	3,27	2,78	VCI/C PIZARRO SP-30SB PC-BON/GR	BIMZ II/2	0706
1	7,13	3,27	2,78	VCI/C PIZARRO SP-30SB PC-BON/GR	BIMZ II/2	0706
1	7,13	3,27	2,78	VCI/C PIZARRO FASE II	BIMZ II/2	0706
1	7,13	3,27	2,78	VCI/C PIZARRO FASE II	BIMZ II/2	0706

15 **EMPLEO, CARGO, FECHA**

FIRMA Y SELLO

IDENTIFICACIÓN DE QUIEN FIRMA

SR. JEFE DE CG BRI X

ANEXOS

T-500

ANEXO PT 50020648-19-530135

DESCRIPCIÓN DEL RECURSO (CONTINUACIÓN)

Nº ELEMENTOS	LARGO (M)	ANCHO (M)	ALTO(M)	RECURSO		CÓDIGO
1,00	7,13	3,27	2,78	VCI/C PIZARRO FASE II	BIMZ II/2	
1,00	6,15	2,57	2,88	VEC 3562.03 M1 TC-25	GACAC II/10	
1,00	6,15	2,57	2,88	VEC 3562.03 M1 TC-25	GACAC II/10	

ANEXOS

T-500

OT | 16 | ACUSE DE RECIBO (A/R) / ACCIÓN (AC)

A/R 3 Sep 19	A/R	A/R
AC X X OT (TÁCHESE LO QUE NO PROCEDA)	AC (TÁCHESE LO QUE NO PROCEDA)	AC (TÁCHESE LO QUE NO PROCEDA)
(FIRMA Y SELLO)	(FIRMA Y SELLO)	(FIRMA Y SELLO)
RAUL SANTIAGO RODRIGUEZ		

DENEGACIÓN, MOTIVOS

OT	17 EMPLEO, CARGO, FECHA	57220450	Tf / Fx / Tx	18 Nº DE ORDEN	19 FECHA
	MALE_DIRECCION. POC ANALISTA 04; 880-6402			50020648-19-9N0135	3 Sep 19

20 RECORRIDO		21 MEDIOS	22 A REALIZAR POR	23 ESCOLTAS	24 CARBURANTE A CARGO DE
20.1 EL HIGUERON	- ZARAGOZA	21.1 FRM	22.1 RENFE OPERADORA		24.1
20.2 ZARAGOZA	- EL HIGUERON	21.2 FRM	22.2 RENFE OPERADORA		24.2
20.3		21.3	22.3		24.3
20.4		21.4	22.4		24.4
20.5		21.5	22.5		24.5
20.6		21.6	22.6		24.6
20.7		21.7	22.7		24.7
20.8		21.8	22.8		24.8
20.9		21.9	22.9		24.9
20.10		21.10	22.10		24.10

25 ÓRGANO QUE ABONA: ET

26 NORMAS DE EJECUCIÓN	27 DESTINATARIOS	
	DE ACCIÓN	DE INFORMACIÓN
AP.10 EME PARA CESET	* EST. FF.CC. EME * RENFE OPERADORA	CGFT CGFLO PLMM RPEI 12

28 AUTORIDAD ORDENANTE — FIRMA Y SELLO

EMPLEO, CARGO, FECHA

IDENTIFICACIÓN DE QUIEN FIRMA DIRINFULOG — 3 Sep 19

29 DILIGENCIAS POSTERIORES, EMPLEO, CARGO Y FIRMA

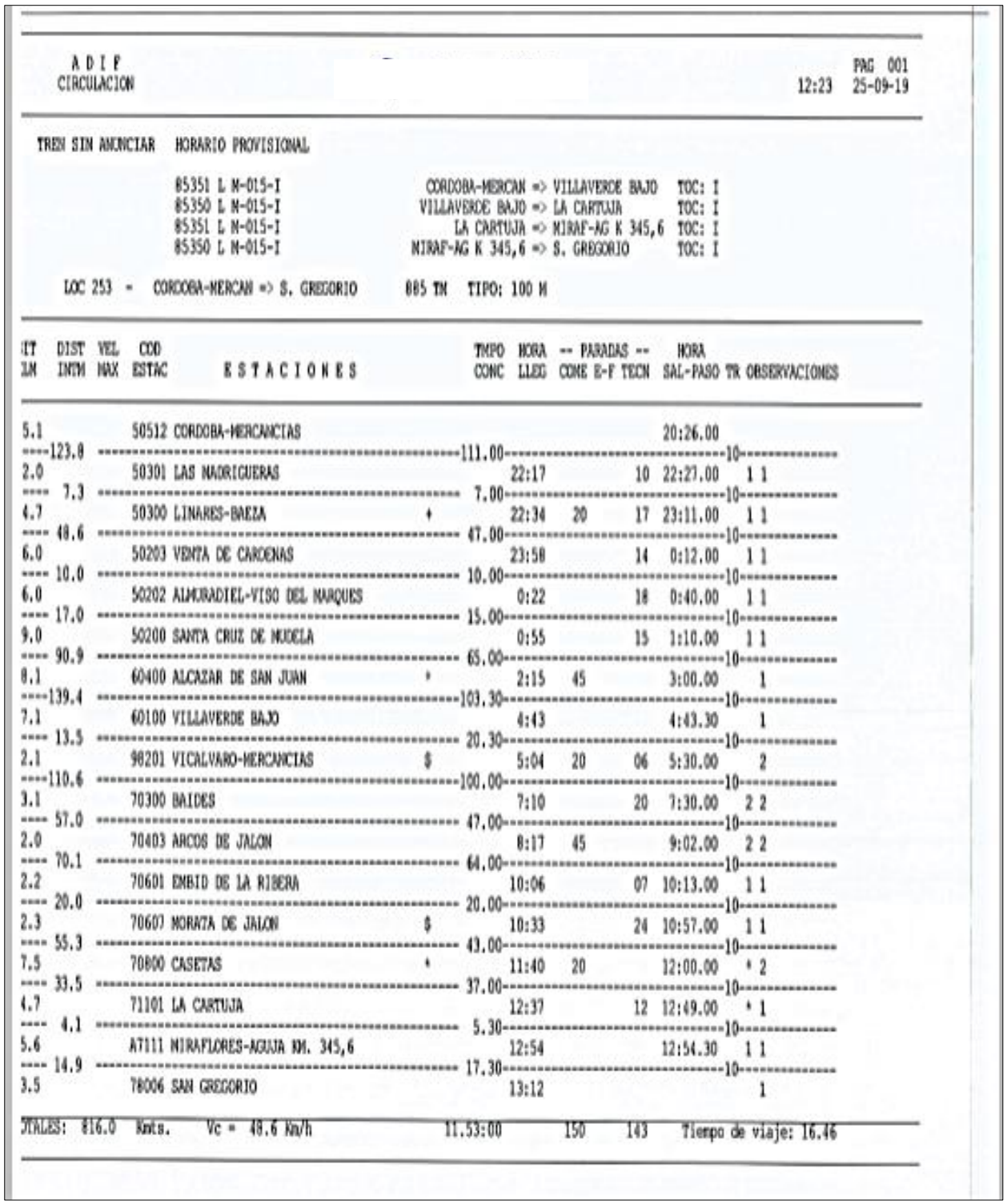
30 CONTROL Y LIQUIDACIÓN DE COSTOS

30.1 MEDIOS MILITARES				30.2 MEDIOS CIVILES	30.3 TOTAL	31 FECHA DE TERMINACIÓN
DIETAS	CARBURANTE	PEAJES	SUMA	SUMA		
0,00	0,00	0,00	0,00	0,00	0,00	

32 REGISTRO INFORMÁTICO. NOMBRE, RÚBRICA Y SELLO

ANEXOS

Apéndice 4. Orden de marcha.



A D I F
CIRCULACION

PAG 001
12:23 25-09-19

TREN SIN ANUNCIAR HORARIO PROVISIONAL

85351 L M-015-I CORDOBA-MERCAN => VILLAVERDE BAJO TOC: I
85350 L M-015-I VILLAVERDE BAJO => LA CARTUJA TOC: I
85351 L M-015-I LA CARTUJA => MIRAF-AG K 345,6 TOC: I
85350 L M-015-I MIRAF-AG K 345,6 => S. GREGORIO TOC: I

LOC 253 - CORDOBA-MERCAN => S. GREGORIO 885 TN TIPO: 100 M

IT IM	DIST INTM	VEL MAX	COD ESTAC	ESTACIONES	TMPO CONC	HORA LLEG	-- PARADAS -- COME	E-F	TECN	HORA SAL-PASO	TR	OBSERVACIONES
5.1			50512	CORDOBA-MERCANCIAS						20:26.00		
	123.0				111.00						10	
2.0			50301	LAS MADRIGUERAS		22:17			10	22:27.00	1	1
	7.3				7.00						10	
4.7			50300	LINARES-BAEZA		22:34	20		17	23:11.00	1	1
	48.6				47.00						10	
6.0			50203	VENTA DE CARDENAS		23:58			14	0:12.00	1	1
	10.0				10.00						10	
6.0			50202	ALMORADIEL-VISO DEL MARQUES		0:22			18	0:40.00	1	1
	17.0				15.00						10	
9.0			50200	SANTA CRUZ DE MUDELA		0:55			15	1:10.00	1	1
	90.9				65.00						10	
0.1			60400	ALCAZAR DE SAN JUAN		2:15	45			3:00.00		1
	139.4				103.30						10	
7.1			60100	VILLAVERDE BAJO		4:43				4:43.30		1
	13.5				20.30						10	
2.1			98201	VICALVARO-MERCANCIAS		5:04	20		06	5:30.00		2
	110.6				100.00						10	
3.1			70300	BAIDES		7:10			20	7:30.00	2	2
	57.0				47.00						10	
2.0			70403	ARCOS DE JALON		8:17	45			9:02.00	2	2
	70.1				64.00						10	
2.2			70601	EMBID DE LA RIBERA		10:06			07	10:13.00	1	1
	20.0				20.00						10	
2.3			70607	MORATA DE JALON		10:33			24	10:57.00	1	1
	55.3				43.00						10	
7.5			70800	CASETAS		11:40	20			12:00.00	*	2
	33.5				37.00						10	
4.7			71101	LA CARTUJA		12:37			12	12:49.00	*	1
	4.1				5.30						10	
5.6			A7111	MIRAFLORES-AGUJA KM. 345,6		12:54				12:54.30	1	1
	14.9				17.30						10	
3.5			78006	SAN GREGORIO		13:12						1
TOTALES: 816.0 Kms. Vc = 48.6 Km/h					11.53:00		150		143	Tiempo de viaje: 16.46		

ANEXOS

ANEXO D. MANUAL DE USUARIO DE LA APLICACIÓN.

(Página siguiente)

MANUAL TÉCNICO

APP PARA LA CARGA DE TRENES MILITARES: LECTURA AUTOMATIZADA DE MATERIALES

TRABAJO DE FIN DE GRADO

AUTOR: PATRICIA DÍAZ AGUILAR

ACADEMIA GENERAL MILITAR
CURSO 2019/2020

CARGA DE TRENES MILITARES

MANUAL TÉCNICO

MANUAL PARA USO INTERNO DE LA CÉLULA DE CONTROL DE MOVIMIENTO POR FERROCARRIL

Contenido

Lista de figuras

1. Introducción

Este manual busca explicar paso a paso cada una de las operaciones que se puede realizar en la aplicación. Se trata de un manual muy gráfico, con lenguaje de fácil interpretación y se construye de tal manera que sea de fácil comprensión para los usuarios finales de la aplicación con el objetivo final de que el usuario irresponsable de este proyecto sea capaz de reconocer fácilmente los aspectos, características y funciones que les proporciona la aplicación.

1.1. Objetivos abordados

Este ha sido el resultado del Trabajo de Fin de Grado cuyo objetivo era llevar a cabo el desarrollo de una aplicación para dispositivos Android que simplifique al máximo el proceso de carga de un tren militar. Para ello, se ha empleado la lectura de un código QR asignado a cada material/vehículo, el cual recopilará las características de estos, para que la aplicación confeccione automáticamente en formato Excel los cuadros de carga y la tasación del tren conforme las tarifas establecidas con la operadora RENFE.

El proceso de carga de todo el material en el tren abarca un trabajo manual muy laborioso realizado por los responsables de operaciones de cargas, por lo que es interesante tratar de automatizar el proceso asignando diferentes códigos QR a cada material. De esta forma, mediante la lectura de dicho código, se registrarán todos los datos relacionados con el mismo (peso, dimensiones, código de identificación, etc). Además, esta aplicación solventa el problema de reconocimiento visual de determinados materiales, ya que las variaciones entre algunos modelos/versiones son reducidas, pero pueden generar alteraciones en el documento generado (de coste principalmente).

En definitiva, se trata de una aplicación para la comprobación de las peticiones de transporte planificadas previamente en un tiempo mucho más reducido y disminuyendo el error humano, dejando como resultado un proceso de carga más eficaz, facilitando, además, en caso de incidencias, que estas sean solventadas con la mayor brevedad posible.

2. Instalación

Para la instalación de la aplicación en cualquier dispositivo Android, se procederá a la instalación de esta mediante un archivo APK (Android Aplication Package). Un archivo APK es un formato utilizado para la instalación de software en Android y permite la instalación de aplicaciones cuando no las encontramos disponibles de otra forma (Google Play, Amazon, etc.). El archivo APK contiene la aplicación en sí misma y el instalador que permite guardarla y ejecutarla en el dispositivo terminal.

Lo primero que hay que hacer previamente a descargar cualquier archivo APK, es activar la opción de instalar aplicaciones que procedan de orígenes desconocidos en el propio dispositivo. Para ello, hay que ir a los ajustes del teléfono, buscar la opción de seguridad y ahí dentro estará la opción de consentir que el dispositivo instale aplicaciones de fuentes desconocidas. Después de hacer esto podemos instalar la aplicación ejecutando el APK y, además, para un uso eficiente de la aplicación tendremos que acceder a los permisos de la misma desde los ajustes del dispositivo y activarlos. De esta forma, se puede comenzar a disfrutar de la app *CARGA PARA TRENES MILITARES* abriéndola desde el menú de aplicaciones del dispositivo.

Ilustración 1. Icono de la aplicación.

3. Herramientas empleadas.

3.1. Android Studio.

El desarrollo de este proyecto se ha llevado a cabo con la plataforma de Android Studio, esta se trata de la herramienta oficial y gratuita de Google para desarrollar aplicaciones en teléfonos móviles Android.

Esta plataforma cuenta además con numerosas ventajas a la hora de programar por particularidades como: ofrecer la corrección de errores en línea y proporcionar una rápida compilación, además de tener todas las herramientas necesarias para programar en cualquier tipo de plataforma y/o dispositivo Android, proporciona la posibilidad de ejecutar la aplicación mediante el uso de emuladores (creando un dispositivo virtual) o directamente conectando el móvil al puerto USB del dispositivo.

Ilustración 2. Logo de Android Studio

3.2. Java.

Java es uno de los lenguajes de programación más ampliamente establecidos. Se ha optado por el empleo de este puesto que es uno de los lenguajes oficiales para el desarrollo de aplicaciones móviles en Android, junto con Kotlin que, a pesar de que está adquiriendo una creciente importancia por su productividad, practicidad y eficacia, la comunidad y los recursos disponibles (librerías, foros de ayuda etc.) todavía no están tan maduros en comparación con Java.

Entre las características más destacables de Java encontramos que es un lenguaje orientado a objetos, es muy flexible; funciona en cualquier plataforma, dado que las aplicaciones que se desarrollan con este lenguaje funcionan en cualquier entorno; su uso no acarrea inversiones económicas, ya que no se precisa de ninguna licencia para programar; es de fuente abierta , lo que quiere decir que ofrece libre acceso a sus librerías nativas para sacar provecho de ellas y/o desarrollarlas; y es un lenguaje expansible, dado que es trabajado por una amplia comunidad de usuarios que ofrecen mejoras y soluciones constantemente.

Ilustración 3. Logo Java.

3.3. DB Browser for SQLite.

La base de datos se ha creado en un primer momento en un archivo Excel, siendo cada una de las hojas que lo componen una tabla de la base de datos. Posteriormente se guardaron con la extensión *.csv* (formato abierto sencillo con valores separados con comas) para ser importados, siendo previamente transformados al formato *.db* mediante el software *DB Browser for SQLite.*

SQLite es una biblioteca de C que implementa un motor de base de datos SQL de código abierto y, debido a que es de tamaño reducido es muy utilizable en los dispositivos móviles y viene por defecto integrada en el sistema Android.

Ilustración 4. Logo de DB Browser for SQLite

4. Funcionalidades.

Para explicar la funcionalidad de la aplicación se ha usado una herramienta bastante gráfica y fácil de comprender que se usa comúnmente en los sistemas orientados a objetos. Se trata del modelo de casos de uso. Este es un tipo de notación UML (*Unified Modeling Language* en inglés) o "lenguaje unificado de modelado" que sirve para especificar y describir los métodos y/o procesos que permiten modelar el comportamiento de un sistema, concretamente identificando los requisitos funcionales del mismo.

En el siguiente diagrama se muestran los casos de uso asociados al actor dentro del sistema. Estos a su vez están relacionados mediante líneas de comunicación que definen la interacción usuario-sistema con dicha funcionalidad y además muestra la relación, en caso de que la haya,

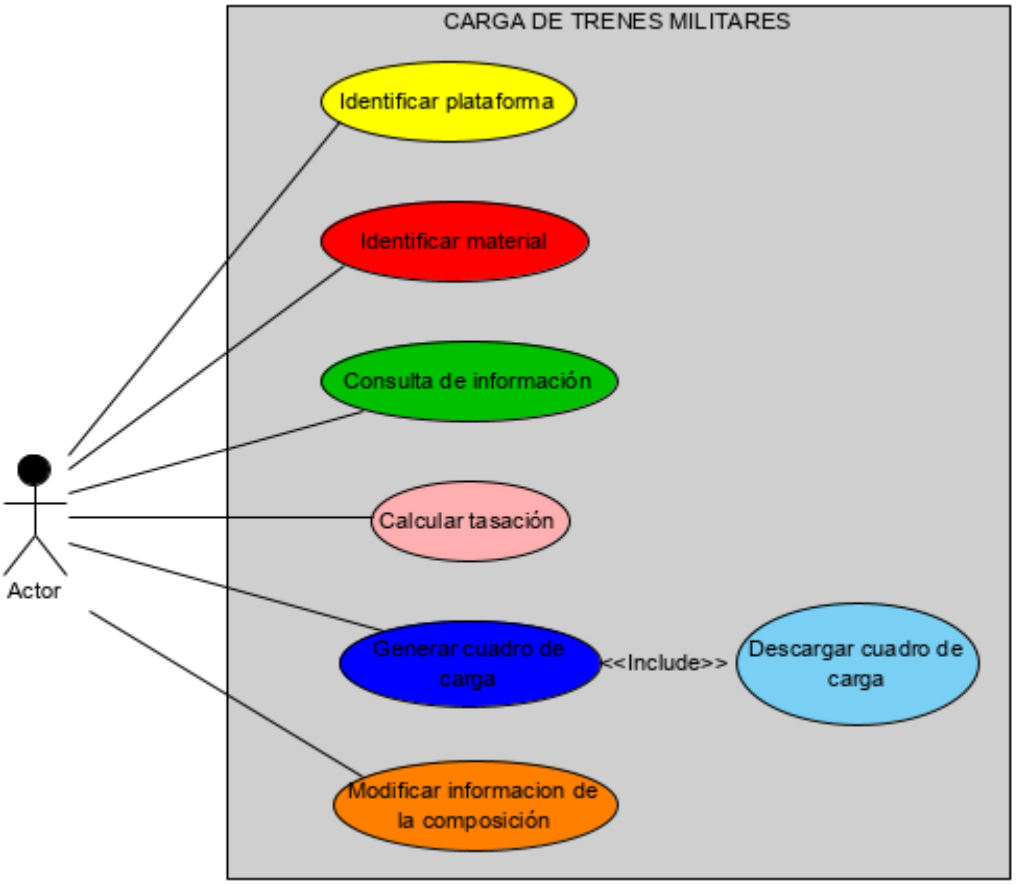


Ilustración 5. Diagrama de casos de uso de la aplicación.

Una vez identificado cual será nuestro entorno del sistema, es decir, el software que vamos a representar, que en este e caso es la aplicación "*CARGA DE TRENES MILITARES*", identificamos al actor externo que interactúa con el sistema, siendo este el ROC y los casos de uso, representados por un conjunto de funcionalidades que el sistema proporciona a los actores y a las que pueden acceder libremente desde este.

- **Identificar plataforma**. El usuario puede obtener la información de una plataforma tras leer su respectivo código QR. Posteriormente, obtiene los siguientes datos de esta: código identificador UIC y tipo de plataforma.

- **Identificar material**. Del mismo modo que el caso anterior, tras leer el código qr del material correspondiente, en pantalla se muestra de forma inmediata la denominación táctica del material y su peso.
- **Consultar información**. El usuario puede acceder a información a través del menú de la aplicación. De esta forma, puede consultar los planos de amarres y las fichas de gálibo de los materiales, las normativas que se aplican en el transporte por ferrocarril (convenio MINISDEF-RENFE y consigna C-41) y las prescripciones particulares de los materiales afectados por las citadas normativas.
- **Calcular tasación del tren**. Una vez que se genera el cuadro de carga, en el fichero descargado podemos obtener el valor de la facturación del tren.
- **Modificar información de la composición**. El usuario puede modificar la información de la composición, actualizando el listado de plataformas y los cargamentos que las ocupan.
- **Generar cuadro de carga.** Una vez almacenada la información necesaria de la composición del tren, el usuario puede generar el cuadro de carga, cuyo resultado será un fichero con extensión ".csv" en el que se muestran las plataformas del tren y el material embarcado en estas en el orden que el usuario las ha ido leyendo. Además, el fichero contiene información adicional de ambos elementos respectivamente.
- **Descargar cuadro de carga**. Este caso de uso contiene una relación de inclusión con el caso de uso anterior, lo que quiere decir que siempre que se genere un cuadro de carga, este automáticamente procede a su descarga.

5. Características.

5.1. Secuencia paso por paso (EJEMPLIFICADA).

La ejemplificación del uso de la aplicación se llevará a cabo mediante el caso del transporte realizado los días 9 y 10 de octubre de 2019, el cual consistía en la ejecución de un ejercicio de transporte de vehículos y material procedentes de la BRI "GUZMÁN EL BUENO" X por el itinerario entre las terminales ferroviarias de Córdoba (El Higuerón) y la estación de San Gregorio (Zaragoza). Se va a utilizar la siguiente composición para clarificar su funcionamiento siguiendo la siguiente secuencia. Posteriormente, se expondrán distintas consideraciones que tiene en cuenta a aplicación de forma que maximice la prevención de errores en el proceso de carga.

NUMERACIÓN (UIC)	CONTENIDO
507105080034	Z118004
837139710037	CC LEOPARDO 2E
837139710052	CC LEOPARDO 2E
837139710102	CC LEOPARDO 2E
837139710086	CC LEOPARDO 2E
837139710383	CC LEOPARDO 2E
837139710466	CC LEOPARDO 2E
837139710532	VCI/C PIZARRO FASE II
837139710540	VCI/C PIZARRO FASE II
837139710375	VCI/C PIZARRO FASE II
837139710482	VCI/C PIZARRO PC BON
837139710516	VCI/C PIZARRO PC BON
837139710284	2 TOA M113 MCCLA TOW
837139710169	CC M47 ER3

Tabla 1. Composición de uno de los dos trenes del transporte.

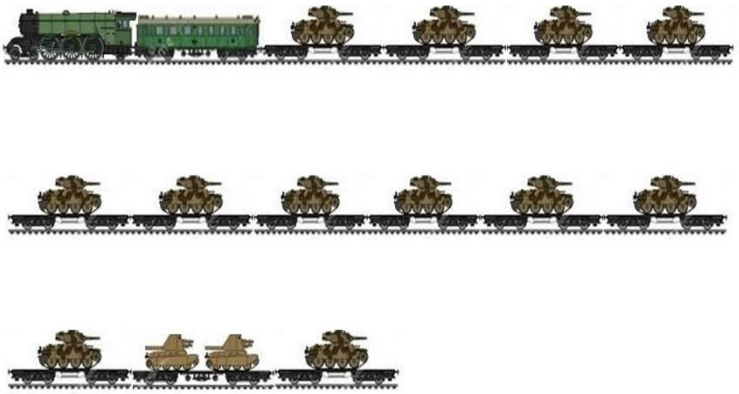


Ilustración 6. Composición grafica del tren

A continuación, se va a mostrar lo que sería el proceso de carga de este tren, mostrando la lectura plataforma por plataforma de esta misma y su contenido. Los respectivos códigos QR se adjuntar en una leyenda para que el usuario pueda probar la aplicación particularmente.

1. Iniciamos la aplicación y pulsamos botón "EMBARQUE". Comprobaremos que nos dirige a la pantalla principal de la aplicación.

Ilustración 7. Pantalla de inicio

Ilustración 8. Estado inicial de la pantalla principal

2. Seleccionamos el botón de Plataforma (aunque ya está programado para que este seleccionado por defecto desde un primer momento).
3. Pulsamos el botón de escanear, la aplicación nos redirige a la cámara, mostrando un marco en el que “encajar” nuestro código a leer.
4. Una vez leído este, veremos cómo en pantalla nos cambia el texto “debe escanear una plataforma” por el código UIC de la plataforma escaneada y el tipo de esta y, bajo la misma, un texto que comunica al usuario el contenido actual de esa plataforma. En este caso la primera plataforma de la composición es el coche de viajeros. Como podemos comprobar, muestra su correspondiente código y tipo, y notifica que la plataforma va vacía en ese instante (“ningún cargamento”).

Ilustración 9. Ubicación botones de plataforma y escanear.

Ilustración 10. Pantalla del escáner (cámara).

Ilustración 11. Información de la plataforma 1 (coche de viajeros).

En este caso, esta plataforma se considerará como “vacía”, debido a que no es ocupada por ningún tipo de cargamento.

5. Pulsamos el botón "+" (situado a la derecha del texto que muestra Plataforma 1 en este momento) para añadir una nueva plataforma a nuestra composición. Podemos observar cómo ha cambiado el índice de la plataforma 1 por "Plataforma 2 de 2" y aparecen las flechas en sus laterales, así como el icono de la papelera, para poder consultar y modificar la información de la plataforma ya escaneada anteriormente.
6. Procedemos de nuevo a escanear una plataforma. Como ya se ha dicho anteriormente, ha de seleccionarse el botón de plataforma.

Ilustración 13. Ubicación botón añadir plataforma.

Ilustración 13. Plataforma añadida.

7. Por el contrario, veamos qué pasa si la aplicación detecta que está seleccionado el botón de material. En la siguiente figura podemos observar que el sistema avisa al usuario de que el elemento que debe escanear es una plataforma.

Ilustración 15. Mensaje al usuario cuando pretende escanear material sin haber una plataforma registrada.

Ilustración 15. Información plataforma 2.

8. Siguiendo con el proceso de embarque, vamos a leer el material que contiene la plataforma. Ha de seleccionarse el botón "material" y posteriormente el de escanear. Este paso ha de repetirse las veces que sea necesario, es decir, si mi plataforma contiene dos mercancías, se comprueba que este el botón de material seleccionado y se procede a la lectura de su código QR correspondiente. De esta forma, ahora nos aparece:

- Un listado con los cargamentos leídos (su denominación táctica, su peso y la opción de eliminarlo de la lista).
- El texto debajo del código y tipo de plataforma se actualiza en consonancia con los materiales escaneados, mostrando en pantalla el número total de materiales que contiene en ese momento la plataforma, así como la suma del peso de estos.

En este caso, vamos a leer la primera plataforma cargada, por lo que nos tiene que aparecer en pantalla los siguientes datos:

- 837139710037-PMMER
- 1 cargamento – 63 Tm (en verde, dado que no excede el peso máximo que soporta la plataforma PMMER)
- CC LEOPARDO 2E- 63 Tm y su correspondiente botón de borrado.

Ilustración 16. Información de la plataforma 2.

9. Para continuar con la lectura de la composición habría que ejecutar de nuevo desde el paso 6 al 9, es decir, añadimos una nueva plataforma, leemos su código QR, y comenzamos a leer el material que la ocupa. De esta forma, este sería el resultado que se obtendría de cada una de ellas:

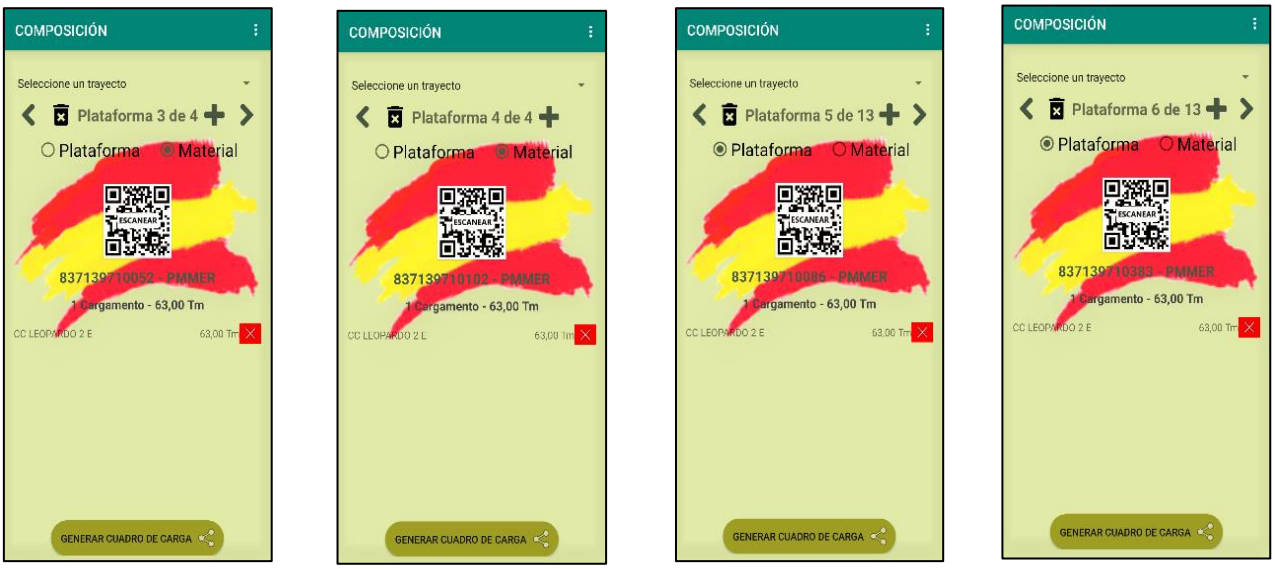


Ilustración 17. Información de las plataformas 3, 4, 5 y 6.

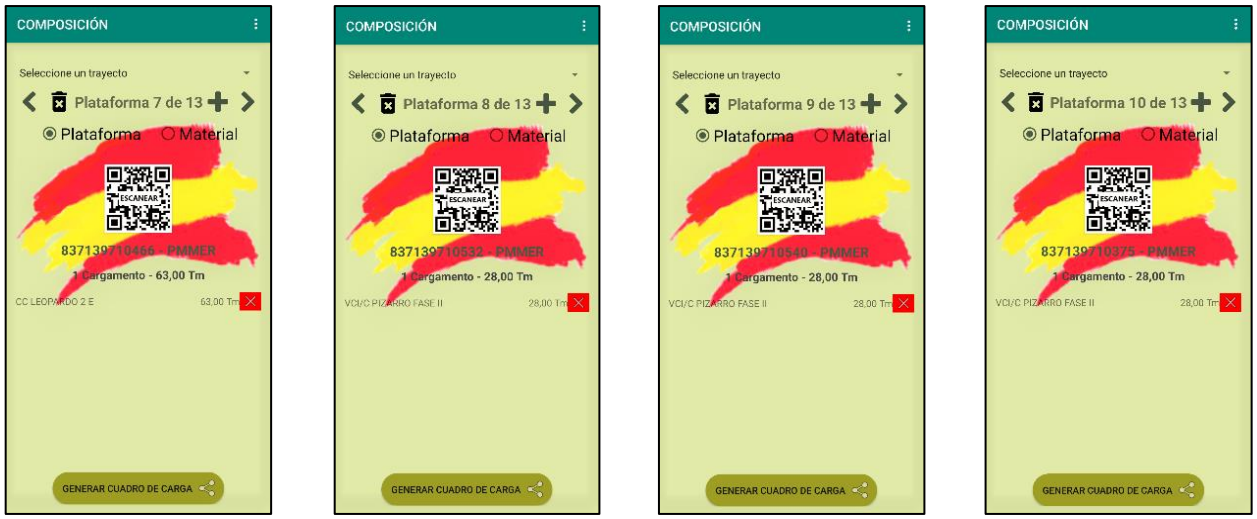


Ilustración 19. Información de las plataformas 7,8 9 y 10.

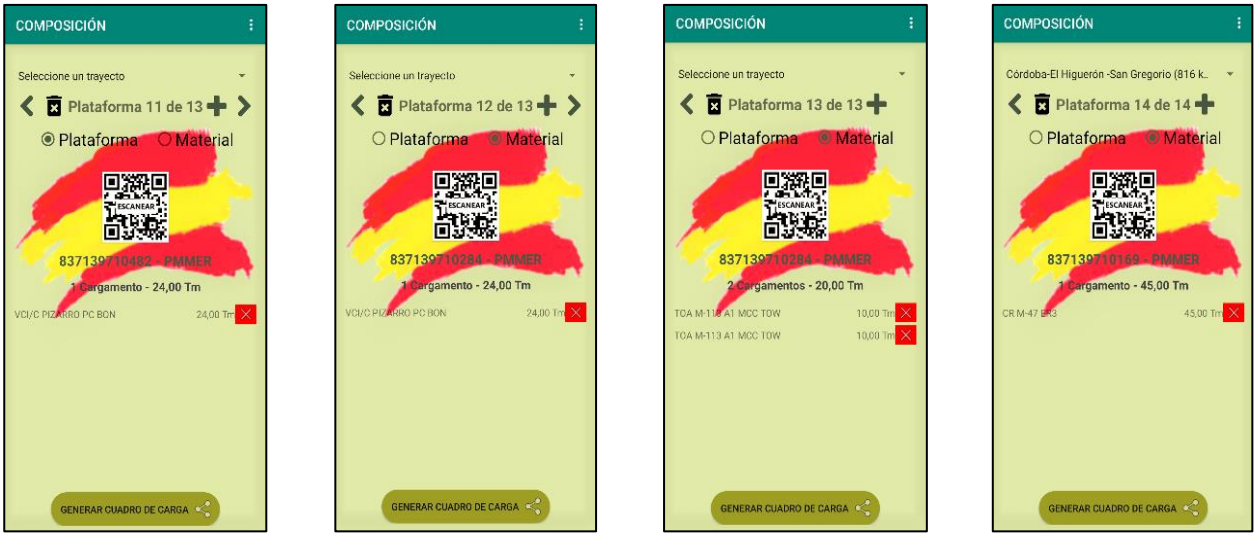


Ilustración 18. Información de las plataformas 11, 12, 13 y 14.

10. Una vez el usuario ha terminado de registrar toda la información de la composición, se procede a generar el cuadro de carga. Para ello se ha de tener en cuenta que:

- **No se hayan añadido plataformas con el botón "+" sin haber almacenado ningún tipo de información en ella**, puesto que se pueden crear infinitas "estructuras" de una plataforma, pero la aplicación detecta si no se tienen datos sobre ella. De esta forma, el sistema pregunta al usuario si las quiere conservar para escanear sus datos o bien desecharlas y no tenerlas en cuenta a la hora de generar el cuadro de carga.
- Ha de haber un **trayecto seleccionado**. Esto se hace pulsando en la lista desplegable nombrada como "Seleccione un trayecto". En caso de que no esté seleccionado, el sistema lanza un mensaje al usuario avisándole de que debe seleccionarlo.
- Estas consideraciones se exponen en el siguiente apartado.

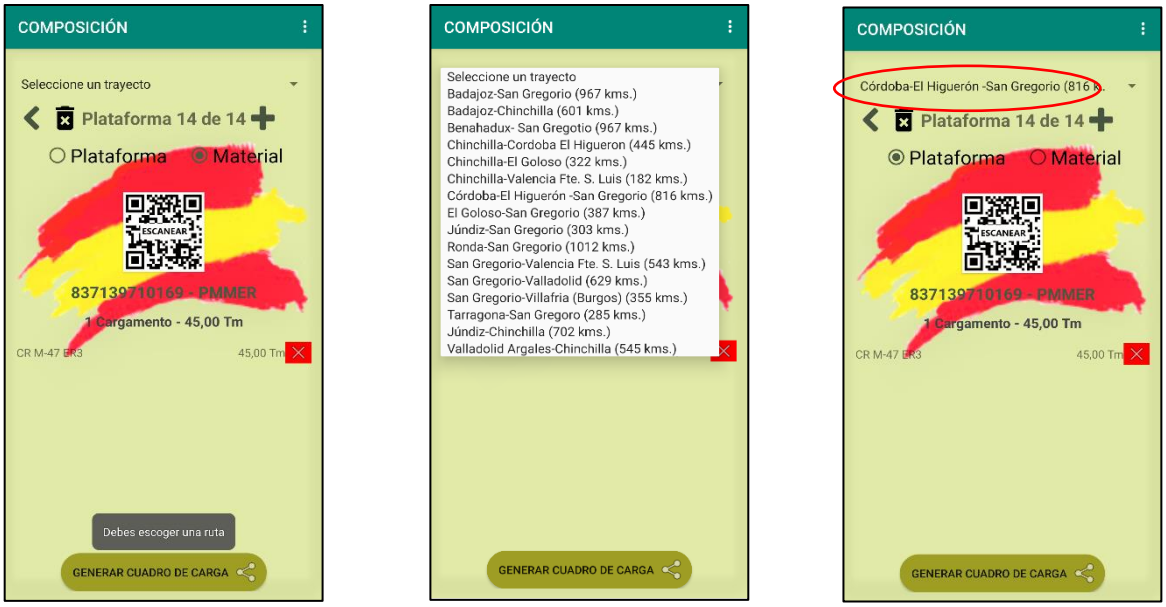


Ilustración 20. Selección del trayecto

11. Una vez se ha generado el cuadro de carga, si deslizamos la barra de estado y herramientas de nuestro dispositivo podemos ver como se está descargando el fichero csv donde ha quedado registrada la información.

12. Para obtener este fichero se ha creado una ruta donde el usuario ya puede operar con el archivo libremente. Esta ruta predefinida es la siguiente:

- Ir a la carpeta propia del dispositivo de descargas, ahí el usuario encontrara una carpeta que recibe el nombre de “Cuadros de carga”, todos los ficheros que se generen desde un dispositivo se almacenaran en esa carpeta de cada uno respectivamente.
- Los archivos quedan nombrados por su trayecto, fecha y hora, de esta forma facilita la diferenciación entre los mismos: “TRAYECTO_FECHA.csv”

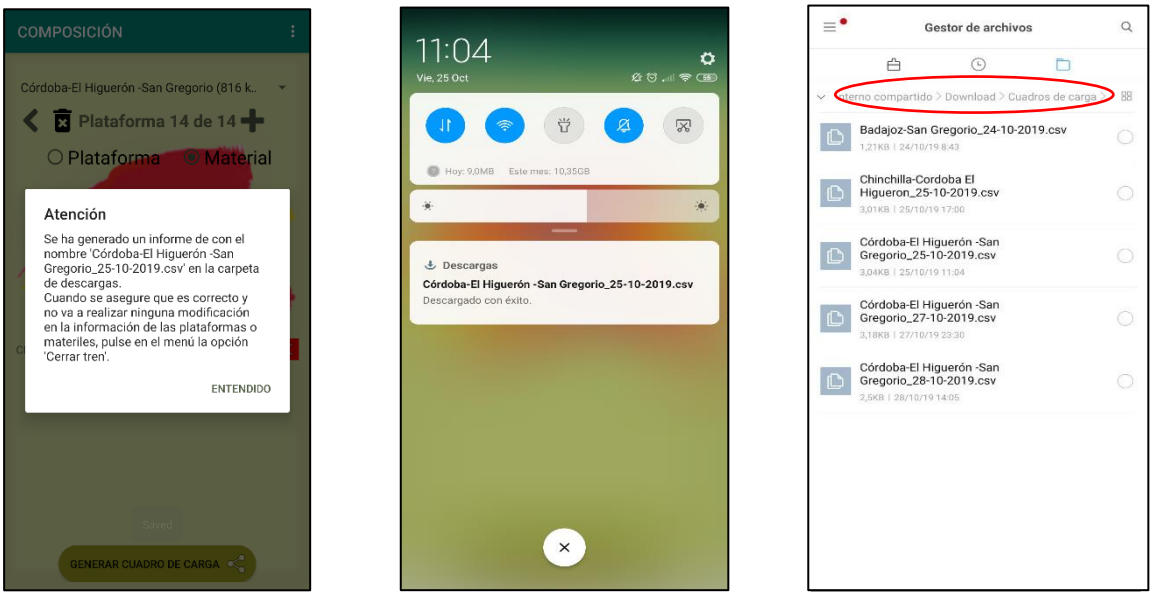


Ilustración 21. Cuadro de carga generado.

13. El fichero final que se obtiene es el siguiente:

TRAYECTO: Córdoba-El Higuerón -San Gregorio

27/10/2019

COMPOSICIÓN

UIC	TIPO	DENOMINACIÓN TÁCTICA	PESO	C-41	ALTO	LARGO	ANCHO	TIPO DE MATERIAL
837139710037	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710052	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710102	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710086	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710383	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710466	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710532	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710540	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710375	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710482	PMMER	VCI/C PIZARRO PC BON	24	null	2932.0	6836.0	3.15	Vehículos blindados
837139710516	PMMER	VCI/C PIZARRO PC BON	24	null	2932.0	6836.0	3.15	Vehículos blindados
837139710284	PMMER	TOA M-113 A1 MCC TOW	10	null	2.45	4.78	2.64	Vehículos blindados
837139710284	PMMER	TOA M-113 A1 MCC TOW	10	null	2.45	4.78	2.64	Vehículos blindados
837139710169	PMMER	CC M-47 ER3	45	X	3.07	8.4	3.62	Carro de combate

TIPO PLATAFORMA	NÚMERO
PMMER	13
BC10X	1
TOTALES	14

DENOMINACIÓN TÁCTICA	NÚMERO	PESO
VCI/C PIZARRO FASE II	3	28.0
VCI/C PIZARRO PC BON	2	24.0
TOA M-113 A1 MCC TOW	2	10.0
CC LEOPARDO 2 E	6	63.0
CC M-47 ER3	1	45.0
TOTALES	14	575.0

COSTE TOTAL (IVA no incluido): 71378.37 €

Ilustración 22. Resultado final de la composición.

Como podemos comprobarnos muestra el trayecto que seguirá el transporte, la fecha en la que se realiza este, la tabla de la composición del tren con sus datos correspondientes (plataforma, tipo, denominación táctica, peso, dimensiones, C-41 y tipo de material), dos tablas resumen de la composición (una para las plataformas y otra para los materiales, agrupados por tipo y nombre respectivamente) y finalmente, el valor final de la tasación del tren.

5.2. Consideraciones.

Las consideraciones que tiene la aplicación se han desarrollado situándonos desde la perspectiva del cargador. De esta forma, la aplicación cuenta con:

- La aplicación cuenta con una sección de documentación. Se puede acceder desde el menú de la aplicación, situado en la esquina superior derecha. Podremos observar el listado de documentos. Entre ellos se encuentran las fichas de gálibo de los materiales, la consigna C-41, las prescripciones de los vehículos afectados por esta última, el convenio MINISDEF-RENFE y otros planos acerca de los amarres de los distintos materiales.

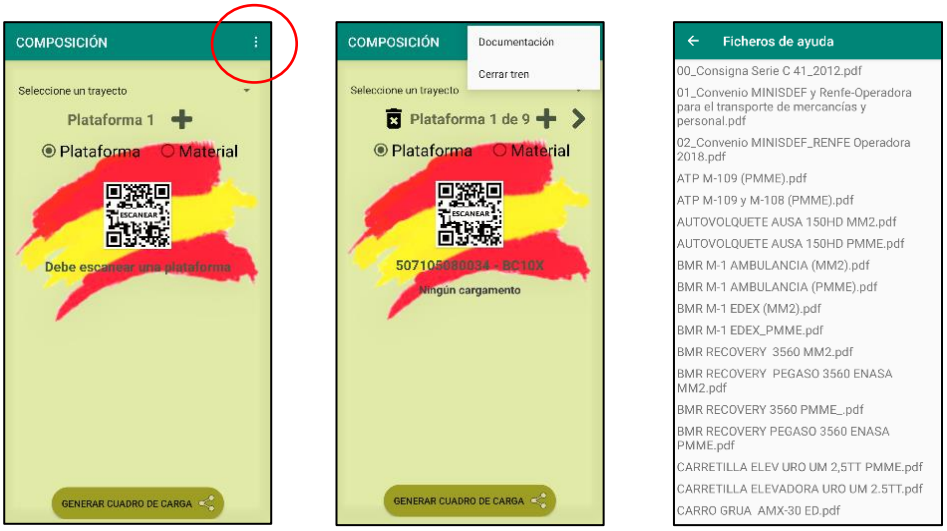


Ilustración 23. Menú superior y documentación con ficheros de ayuda.

- Para poder escanear el material, la aplicación tiene en cuenta que se debe de haber registrado previamente la plataforma donde va a ser cargado. De esta forma, si se selecciona el botón "material" y le damos a escanear sin haber escaneado una plataforma, el sistema nos lo impide, lanzando un aviso al usuario notificando que "Debe escanear una plataforma".

Ilustración 24. Mensaje que lanza la app al usuario cuando este quiere almacenar material sin plataforma registrada.

- El sistema detecta si el usuario quiere leer dos plataformas sucesivamente, encontrándose en la misma posición, es decir, me encuentro en la plataforma X (índice superior), escaneo el código de la plataforma Y, mostrándose en pantalla su código identificativo y tipo. Ahora le doy de nuevo al botón de escanear (estando seleccionado el botón "plataforma" en la misma plataforma X), por lo que la aplicación avisa al usuario sobre el cambio de plataforma solicitando confirmación por parte de este. En caso de aceptar, se lee el nuevo código y se actualiza la información de la plataforma que se encuentra en esa posición.

Ilustración 25. Solicitud para modificar la información de una plataforma ya guardada.

- Como hemos dicho, se puede crear una estructura de plataforma sin necesidad de almacenar información en ella. De esta forma, cuando pulsamos el botón "Generar cuadro de carga", la aplicación detecta si hay una de esas estructuras de plataforma vacías en las que no se ha almacenado ningún dato. El sistema considera estas como plataformas "No válidas", por lo que avisa al usuario y solicita su confirmación para desecharlas, lo que implica que se eliminen automáticamente y no se tengan en cuenta al generar el archivo final.

Ilustración 27. Estructura de plataforma: plataforma sin ninguna información.

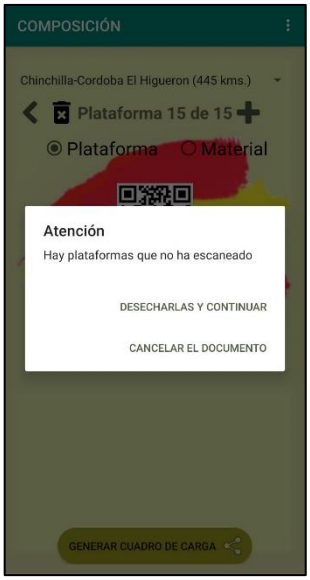


Ilustración 26. Aviso al usuario cuando hay plataformas vacías.

- Si en algún momento se escanea un código que no pertenezca a la base de datos, ya sea de plataforma o material, la aplicación avisa al usuario de que el código no es válido.

Ilustración 28. Mensaje de código no valido.

- Cuando se almacena un listado de materiales dentro de una plataforma, vemos como el texto bajo el código de la plataforma se actualiza en función del número de materiales leídos y la suma de sus pesos. Por ello, el sistema detecta si el peso de los materiales que ocupan una plataforma es superior al máximo peso que esta es capaz de soportar. Por ello, dicho texto cambia de color (se pone en rojo) y no deja generar el cuadro de carga. Para solventar este problema, se puede eliminar el cargamento que se seleccione de la lista pulsando la cruz ubicada en la fila de cada material, y avisa al usuario solicitando su confirmación.

Ilustración 29. Mensaje de peso de plataforma excedido.

Ilustración 30. El sistema pide confirmación para eliminar el cargamento.

- De la misma forma que el cargamento, también se puede eliminar una plataforma. De este modo, se eliminaría toda la información que se hubiese almacenado en ella. Para ello solo hay que seleccionar el botón de "Delete", representado mediante el icono de una papelera. Para borrarla, el sistema pedirá confirmación al usuario.

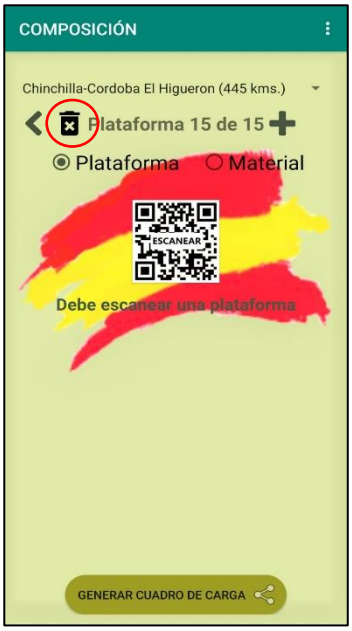


Ilustración 32. Ubicación botón Delete: eliminar plataforma.

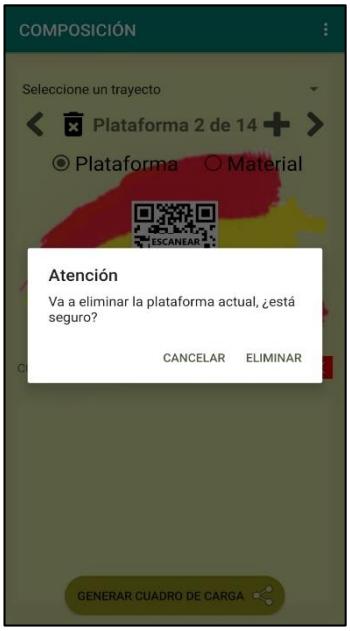


Ilustración 31. El sistema pide confirmación para eliminar la plataforma

- Finalmente, cuando ya se ha registrado toda la información de la composición, al darle al botón de "Generar cuadro de carga", el sistema detecta si el usuario no ha seleccionado ningún trayecto de la lista. De esta forma, la aplicación lanza un mensaje al usuario avisándole de que "seleccione un trayecto". *(Ilustración 20)*

- Una vez este todo correcto al pulsar el botón de "generar cuadro de carga", la aplicación generará el archivo y lo descarga automáticamente. En este caso lanza el mensaje de "Saved" y, para la descarga del fichero final, se ha creado una ruta concreta para facilitar la localización del archivo. Esta es una carpeta que recibe el nombre de "Cuadros de carga" y se encuentra dentro de la carpeta "Downloads" de nuestro dispositivo. El nombre del fichero será el del trayecto, la fecha y la hora (trayecto_fecha.csv). *(Ilustración 21)*

- Por último, para generar un tren nuevo o borrar toda la información que se haya leído hasta el momento. En el menú de la barra superior, a parte de la opción de los documentos PDF (primer punto del presente apartado), tenemos la opción de “Cerrar tren”, lo que implica el borrado de toda la información registrada. Al igual que los casos anteriores, la aplicación avisará y pedirá la confirmación al usuario para el borrado de información almacenada en la memoria del dispositivo.

Ilustración 34. Aviso al usuario de que se va a eliminar la información del tren.

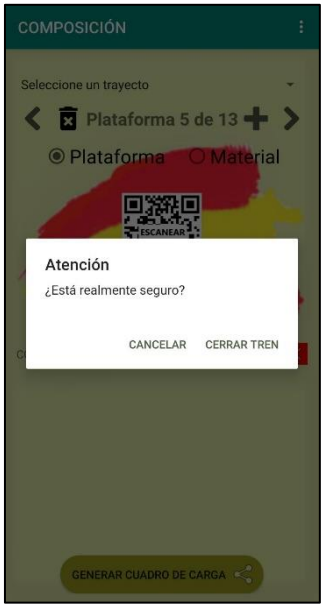


Ilustración 33. El sistema pide confirmación para borrar la información del tren.

CÓDIGOS QR DE LA COMPOSICIÓN DEL TREN

PLATAFORMA (UIC)	507105080034	837139710037	837139710052	837139710102	837139710086	837139710383
CONTENIDO	-	CC LEOPARDO 2E	CC LEOPARDO 2E	CC LEOPARDO 2E	CC LEOPARDO 2E	CC LEOPARDO 2E

PLATAFORMA (UIC)	837139710466	837139710532	837139710540	837139710375	837139710482	837139710516
CONTENIDO	CC LEOPARDO 2E	VCI/C PIZARRO FASE II	VCI/C PIZARRO FASE II	VCI/C PIZARRO FASE II	VCI/C PIZARRO PC BON	VCI/C PIZARRO PC BON

PLATAFORMA (UIC)	837139710284	837139710169
CONTENIDO	2X TOA M113 MCCLA TOW	CC M47 ER3

6. Modificación del sistema.

6.1. Modificación base de datos en SQLite.

En primer lugar, para poder hacer las modificaciones necesarias en la base de datos se requiere tener instalado el programa *DB Browser for SQLite*, el cual se puede encontrar en la siguiente dirección: https://sqlitebrowser.org/dl/.

A continuación, se van a explicar los pasos a seguir para modificar y actualizar la base de datos, los cuales serán representados a posteriori gráficamente mediante imágenes del sistema.

1. Descarga del DB Browser for SQLite. Dentro de las distintas versiones, descargar la que se ajuste a las propiedades del ordenador donde se ejecute la modificación.
2. Abrir el programa. Se debería mostrar como aparece en la *Ilustración 35*
3. Pinchar en el botón de abrir base de datos. (*Ilustración 36)*
4. Buscar en nuestro equipo la base de datos que se ha creado para este proyecto. Debe de aparecer como el icono y el nombre de la *Ilustración 37.*
5. La base de datos nos aparecerá cargada como aparece en la *Ilustración 38*. de esta forma, cambiando de pestaña a la "hoja de datos" veremos, la información de la base de datos almacenada en forma de tabla. Se puede modificar las tablas mediante los botones de "Nuevo Registro", donde se añadirá una fila nueva o "Borrar registro", eliminando la fila seleccionada. (*Ilustración 39)*
6. Los campos a rellenar cada elemento de cada tabla son los que se muestran en la *Ilustración 40.*

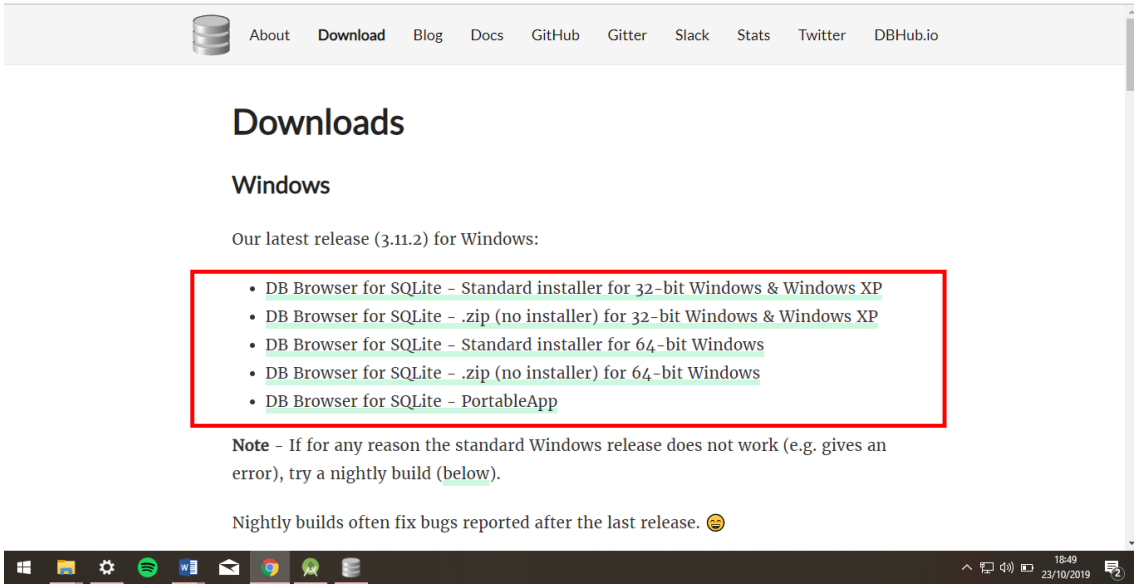


Ilustración 35. Descarga de DB Browser for SQLite

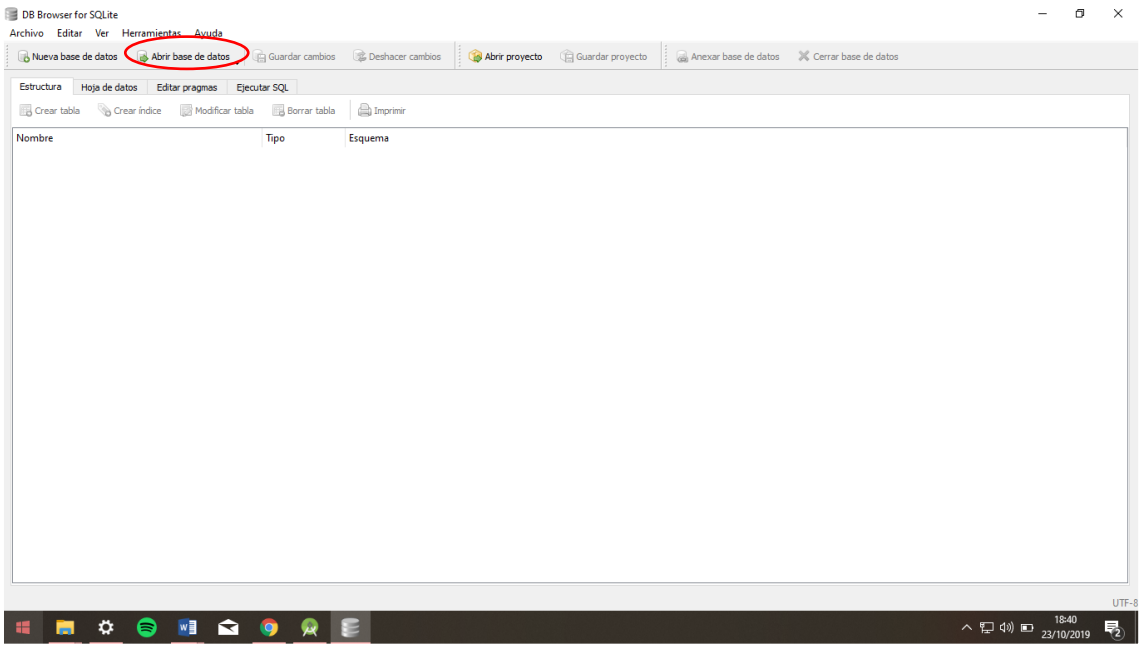


Ilustración 36. Pantalla inicial del programa

Ilustración 37. Icono de la base de datos

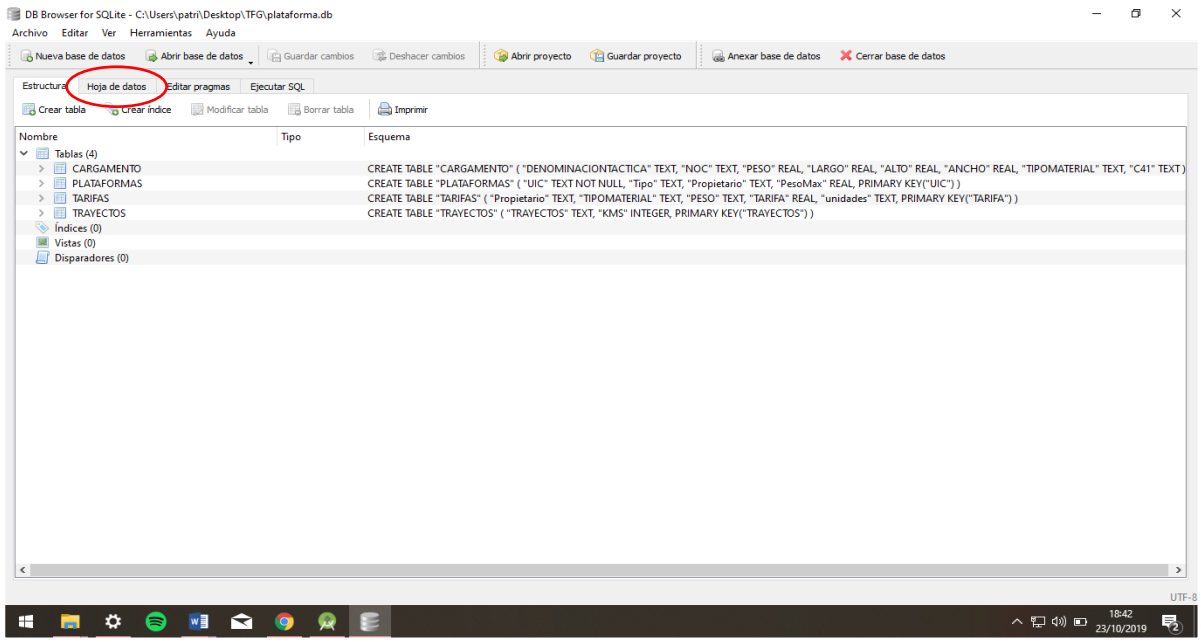


Ilustración 38. Programa una vez abierta la base de datos en el sistema

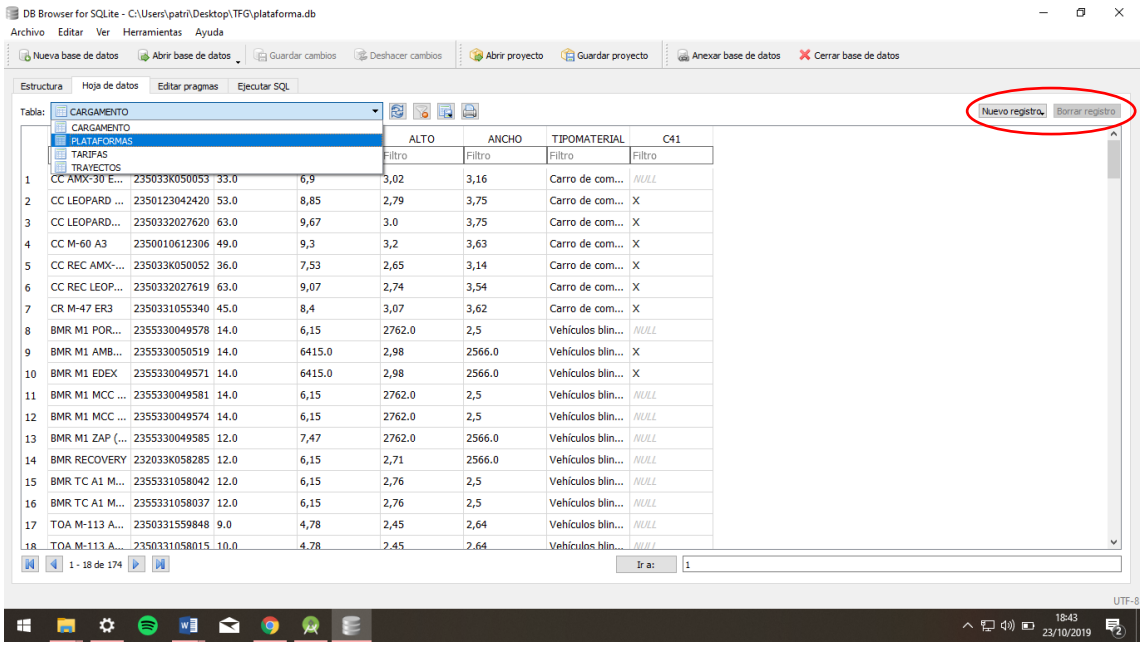


Ilustración 39. Pestaña hoja de datos. Lista de las tablas de la base de datos.

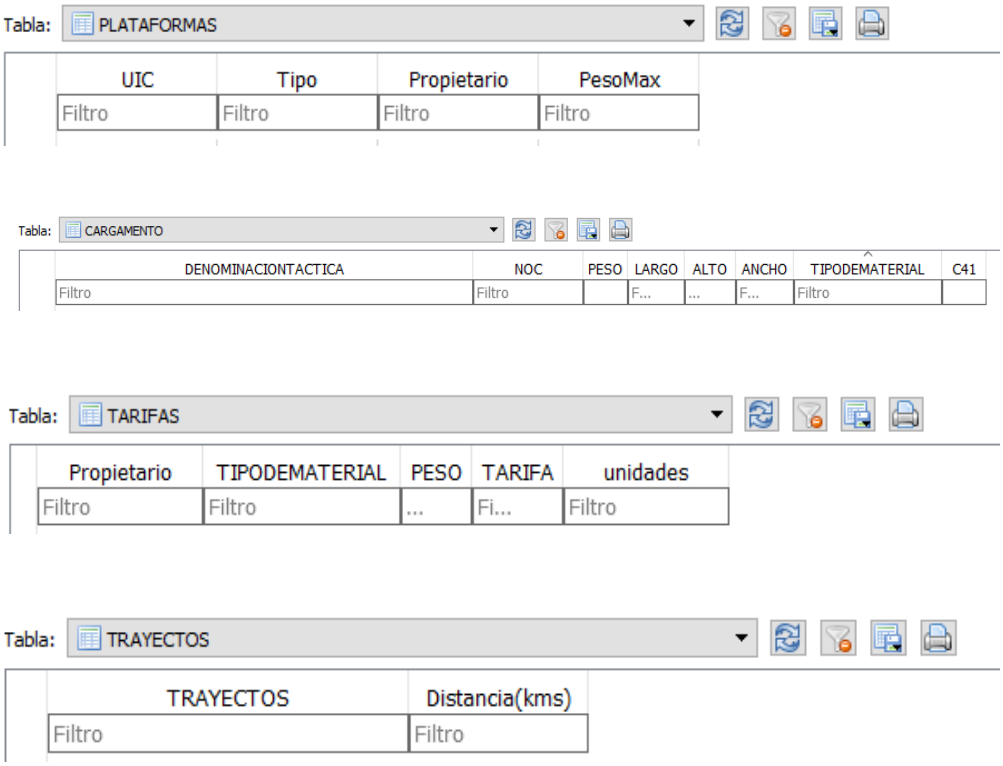


Ilustración 40. Campos de cada tabla de la base de datos.

6.2. Implementación de la base de datos en Android Studio.

Al igual que en el caso anterior, necesitamos instalar *Android Studio* para hacer las pertinentes modificaciones. En este apartado se va a explicar el procedimiento a seguir para actualizar la base de datos con la que trabajará la aplicación. Por lo tanto, del mismo modo que con SQLite, se va a exponer previamente los pasos a seguir, y posteriormente se mostraran de forma gráfica.

1. Descargar JavaScript, dado que este ha sido el lenguaje de programación que se ha empleado para desarrollar la app dentro del entorno Android. Disponible en https://www.java.com/es/download/. (*Ilustración 41)*
2. Descargar Android Studio. Se puede encontrar en la siguiente página: https://developer.android.com/studio/index.html?hl=es-419. (*Ilustración 42*)
3. Una vez descargado y ejecutado en nuestro equipo, abrir el programa. De esta forma se mostrará la pantalla inicial donde seleccionaremos la opción de abrir un proyecto existente. *(Ilustración 43)*
4. Una vez se abra la aplicación, poner el programa en vista Android, como se indica en la *Ilustración 44*. De esta forma, buscaremos en la carpeta *assets* y borraremos la base de datos que contiene la aplicación en ese momento (clic derecho → "Delete") y pegamos en la misma carpeta (*assets*) la base de datos actualizada.
5. El hecho de actualizar la base de datos implica actualizar de la misma forma la aplicación en los dispositivos móviles. En este caso hay que reinstalar la aplicación como se explica anteriormente en el *apartado 2.*

Ilustración 41. Descarga de Java.

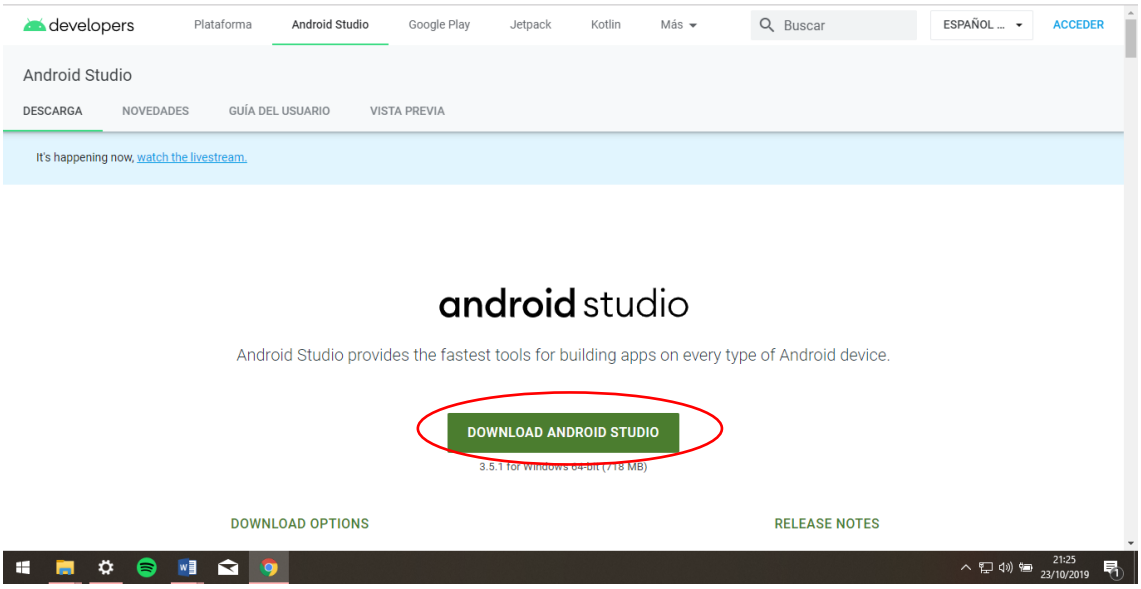


Ilustración 42. Descarga de Android Studio

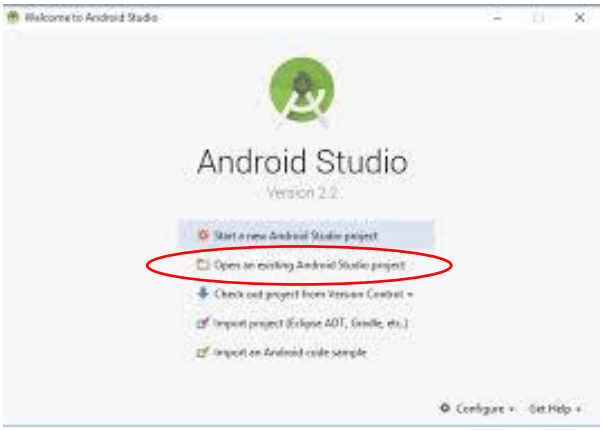


Ilustración 43. Pantalla inicio Android Studio

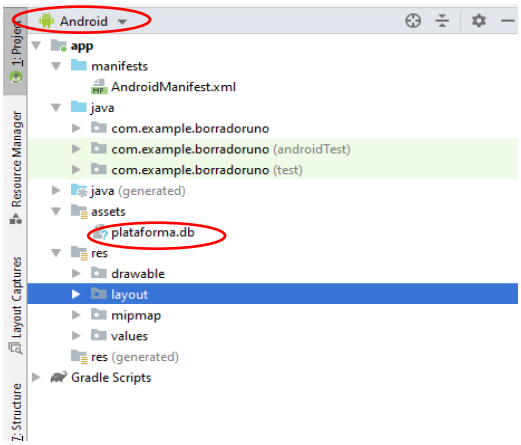


Ilustración 44. Carpeta assets en vista Android.

6.3. Modificación de la información.

Al igual que en el caso anterior, para modificar la información o añadir más documentos PDF al listado, se ha de proceder al igual que para actualizar la base de datos.

1. Ir a la carpeta de *assets.*
2. Eliminar los documentos que se quieran sustituir.
3. Añadir los archivos en la citada carpeta copiando y pegando.

(*) Al igual que el caso anterior, toda modificación del sistema conlleva la reinstalación de este en el dispositivo.

En el caso de la reinstalación mediante el APK, se ha de saber que cada vez que se actualiza el código, se actualiza el APK automáticamente. De esta forma, con ir a la carpeta “app” dentro de nuestro proyecto, posteriormente a la carpeta “release” y copiamos el fichero con extensión .apk en el dispositivo y, por último, desde el gestor de archivos de nuestro dispositivo vamos a la carpeta donde hayamos guardado el archivo APK y lo ejecutamos de nuevo.

ANEXO E. COMPARACIÓN DE RESULTADOS.

Apéndice 1. Peticiones de transporte.

PETICION DE TRANSPORTE

Miercoles, 09 de Octubre de 2019 - Hora: 14.14.36

DATOS DE LA EXPEDICION

Número de Petición: 358232

Número de Expedicion: 17599 Fecha de Expedicion: 20191009

Origen.: 50512 - CORDOBA-MERCANC. Entrega en: Estacion

Destino.: 78006 - SAN GREGORIO Recogida en: Estacion

OTROS DATOS

Codificacion fiscal:	Op. Interna
Contrato:	03301
Fecha de solicitud de entrega DDMMAAAA:	09102019
Numero de Vagones :	12

CONSIGNATARIO:

Cif:	ESS2800811H
Tarjeta:	0143058810
Telefono:	
Nombre:	JEFATURA ASUNTOS ECONOMIC
Direcion:	PRIM, 4 - CUARTEL GENERAL DEL E.T.
e-mail:	

REMITENTE:

Cif:	ESS2800811H
Tarjeta:	0143058810
Telefono:	
Nombre:	JEFATURA ASUNTOS ECONOMIC
Direcion:	PRIM, 4 - CUARTEL GENERAL DEL E.T.

DATOS DE LA EXPEDICION

Remesa (Kg)	0
Mercancia	981028
MMPP.	
NUM. Autos	0
ABCD	0004

DATOS DEL CLIENTE QUE ABONA LA EXPEDICION

Tarjeta:	0143058810
Cif:	ESS2800811H
Telefono:	
Razon social:	JEFATURA ASUNTOS ECON
Domicilio:	PRIM, 4 - CUARTEL GEN
Codigo Postal:	28004
Poblacion:	MADRID

Mensaje: TREN MIILITAR M015I CORD HIG(COR)-S.GREG(ZAR)

VAGONES

Tipo	Num. Vagon	Peso Neto	Mercancia	Merc. Peligr.	Autos	ABCD	Consig.	Remiten.
BBL	507105080059	2.000	981028		0	0007		
PMM	837139710011	63.000	981028		0	0004		
PMM	837139710029	63.000	981028		0	0004		
PMM	837139710060	63.000	981028		0	0004		
PMM	837139710128	63.000	981028		0	0004		
PMM	837139710136	16.000	981028		0	0004		
PMM	837139710144	63.000	981028		0	0004		
PMM	837139710177	63.000	981028		0	0004		
PMM	837139710201	63.000	981028		0	0004		
PMM	837139710219	16.000	981028		0	0004		
PMM	837139710276	63.000	981028		0	0004		
PMM	837139710433	63.000	981028		0	0004		
Total:		**601.000**			**0**			

ANEXOS

PETICION DE TRANSPORTE

Miercoles, 09 de Octubre de 2019 - Hora: 18.37.18

DATOS DE LA EXPEDICION

Número de Petición: 358252

Número de Expedicion: 17623 Fecha de Expedicion: 20191009

Origen.:	50512 -	CORDOBA-MERCANC.	Entrega en:	Estacion
Destino.:	78006 -	SAN GREGORIO	Recogida en:	Estacion

OTROS DATOS

Codificacion fiscal:	Op. Interna
Contrato:	
Fecha de solicitud de entrega DDMMAAAA:	09102019
Numero de Vagones :	14

CONSIGNATARIO:

Cif:	ESS2800811H
Tarjeta:	0143058810
Telefono:	
Nombre:	JEFATURA ASUNTOS ECONOMIC
Direcion:	PRIM, 4 - CUARTEL GENERAL DEL E.T.
e-mail:	

REMITENTE:

Cif:	ESS2800811H
Tarjeta:	0143058810
Telefono:	
Nombre:	JEFATURA ASUNTOS ECONOMIC
Direcion:	PRIM, 4 - CUARTEL GENERAL DEL E.T.

DATOS DE LA EXPEDICION

Remesa (Kg)	**0**
Mercancia	**981028**
MMPP.	
NUM. Autos	**0**
ABCD	**0004**

DATOS DEL CLIENTE QUE ABONA LA EXPEDICION

Tarjeta:	0143058810
Cif:	ESS2800811H
Telefono:	
Razon social:	JEFATURA ASUNTOS ECON
Domicilio:	PRIM, 4 - CUARTEL GEN
Codigo Postal:	28004
Poblacion:	MADRID

Mensaje: TREN MILITAR M017-I DE COR HIGUERON(COR) a S. GREG (ZAR)

VAGONES

Tipo	Num. Vagon	Peso Neto	Mercancia	Merc. Peligr.	Autos	ABCD	Consig.	Remiten.
BBL	507105080034	2.000	981028		0	0007		
PMM	837139710037	63.000	981028		0	0004		
PMM	837139710052	63.000	981028		0	0004		
PMM	837139710086	63.000	981028		0	0004		
PMM	837139710102	63.000	981028		0	0004		
PMM	837139710169	45.000	981028		0	0004		
PMM	837139710284	20.000	981028		0	0005		
PMM	837139710375	28.000	981028		0	0004		
PMM	837139710383	63.000	981028		0	0004		
PMM	837139710466	63.000	981028		0	0004		
PMM	837139710482	24.000	981028		0	0004		
PMM	837139710516	24.000	981028		0	0004		
PMM	837139710532	28.000	981028		0	0004		
PMM	837139710540	28.000	981028		0	0004		
Total:		**577.000**			**0**			

ANEXOS

<u>Apéndice 2</u>. Cuadros de carga generados por el CCMR con los datos proporcionados por el ROC.

ESTACIÓN: CORDOBA EL HIGUERON(COR) - SAN GREGORIO (ZRGZ)

TREN MILITAR N.º: M 015 I

N.º PETICION TRANSPORTE: 358232

N.º EXPEDICIÓN: 17599

FECHA PT: 09/10/19

COMPOSICIÓN
TREN 1

Nº PLAT	<=2tn	> 2tn	Nº CC	TIPO PLATAF	NUMERACIÓN	CONTENIDO	C-41	PESOS				PESO TOTAL
1				BC10X	507105080059	Z118006		2				2
1		1	1	PMMER	837139710029	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710201	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710433	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710011	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710144	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710060	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710276	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710177	CC LEOPARDO 2E	**C-41**	63				63
1		1		PMMER	837139710219	VEC M1		16				16
1		1		PMMER	837139710136	VEC M1		16				16
1		1		PMMER	837139710128	CREC LEOPARDO 2ER “BÚFALO”	**C-41**	63				63
												0
								TOTALES				601

OBSERVACIONES: S/N

leonciom@renfe.es; pgomezm@renfe.es; mercancias24horas@renfe.es; csc.facturacion@renfe.es; opterminales@renfe.es; madridmerc1@renfe.es; segcentro.mercancias@renfe.es sevillamerc1@renfe.es; sevillamerc2@renfe.es

	Nº PLATAFORMAS	PESOS
COCHE VIAJEROS	1	2
PMMER	11	599
MM2		
M-1		
TOTALES	12	601

VEHICULOS		KM
<= 2 TN	0	816
> 2 TN	11	
nº CC	8	

ANEXOS

ESTACIÓN: CORDOBA EL HIGUERON (COR) - SAN GREGORIO (ZAR)

TREN MILITAR N.º: M-017-I

N.º PETICION TRANSPORTE: 358252

N.º EXPEDICIÓN: 17623

FECHA PT: 09/10/19

COMPOSICIÓN
TREN 2

Nº PLAT	<=2tn	> 2tn	Nº CC	TIPO PLATAF	NUMERACIÓN	CONTENIDO	C-41	PESOS				PESO TOTAL
1				BC10X	507105080034	Z118004		2				2
1		1	1	PMMER	837139710037	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710052	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710102	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710086	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710383	CC LEOPARDO 2E	**C-41**	63				63
1		1	1	PMMER	837139710466	CC LEOPARDO 2E	**C-41**	63				63
1		1		PMMER	837139710532	VCI/C PIZARRO FASE II		28				28
1		1		PMMER	837139710540	VCI/C PIZARRO FASE II		28				28
1		1		PMMER	837139710375	VCI/C PIZARRO FASE II		28				28
1		1		PMMER	837139710482	VCI/C PIZARRO PC BON		24				24
1		1		PMMER	837139710516	VCI/C PIZARRO PC BON		24				24
1		2		PMMER	837139710284	2 TOA M113 MCCLA TOW		10	10			20
1		1		PMMER	837139710169	CC M47 ER3	**C41**	45				45
								TOTALES				577

OBSERVACIONES: S/N

leonciom@renfe.es;pgomezm@renfe.es; mercancias24horas@renfe.es; csc.facturacion@renfe.es; madridmerc1@renfe.es; segcentro.mercancias@renfe.es; opterminalesmer@renfe.es; sevillamerc1@renfe.es; sevillamerc2@renfe.es

	Nº PLATAFORMAS	PESOS
COCHE VIAJEROS	1	2
PMME	13	575
MM2		
M-1		
TOTALES	14	577

VEHICULOS		KM
<= 2 TN	0	816
> 2 TN	14	
nº CC	6	

ANEXOS

Apéndice 3. Tasación de los trenes.

TRANSPORTE POR FERROCARRIL DE LA BRI X (Estación de Mercancías de El Higuerón) serie 96

TREN	VEHICULOS	N.º	PMME-ER	MM2	TN, s	KM,s	PRECIO	SUBTOTAL	TOTAL
1	COCHE VIAJEROS	1			2		576.99	576.99	80,140.58 €
	CC LEOPARDO 2E	8	8		504	816	0.1660	68,269.82	
	CREC LEOPARDO 2ER "BUFALO"	1	1		63	816	0.1660	8,533.73	
	VEC M1	2	2		32	816	0.1057	2,760.04	
	SUBTOTAL VH	**11**	**11**	**0**	**569**			80140.58	
TRANSPORTE DE IDA									**80,140.58 €**
TOTAL									**80,140.58 €**
21 % IVA									**16,829.52 €**
TOTAL TRANSPORTE									**96,970.10 €**

TREN	VEHICULOS	N.º	PMME-ER	MM2	TN,s	KM, s	PRECIO	SUBTOTAL	TOTAL
2	COCHE VIAJEROS	1			2		576.99	576.99	71,378.37 €
	CC LEOPARDO 2E	6	14		378	816	0.1660	51,202.37	
	CREC M47 ER3	1	1		45	816	0.1660	6,095.52	
	VCI/C PIZARRO FASE II	3	3		84	816	0.1057	7,245.10	
	VCI/C PIZARRO PCBON	2	2		48	816	0.1057	4,140.06	
	TOA M113 MCCLA TOW	2	1		20	816	0.1298	2,118.34	
	SUBTOTAL VH	**14**	**24**	**0**	**557**			71378.372	
TRANSPORTE DE IDA									**71,378.37 €**
TOTAL									**71,378.37 €**
21 % IVA									**14,989.46 €**
TOTAL TRANSPORTE									**86,367.83 €**

ANEXOS

Apéndice 4. Resultados de la aplicación CARGA DE TRENES MILITARES.

RESULTADO TREN 1.

TRAYECTO: Córdoba-El Higuerón -San Gregorio

28/10/2019

COMPOSICIÓN

UIC	TIPO	DENOMINACIÓN TÁCTICA	PESO	C-41	ALTO	LARGO	ANCHO	TIPO DE MATERIAL
507105080059	BC10X							
837139710029	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710201	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710433	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710011	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710144	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710029	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710276	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710177	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710219	PMMER	VEC M1	16	null	2885.0	6.15	2.5	Vehículos blindados
837139710136	PMMER	VEC M1	16	null	2885.0	6.15	2.5	Vehículos blindados
837139710128	PMMER	CC REC LEOPARDO 2 ER (BÚFALO)	63	X	2.74	9.07	3.54	Carro de combate

TIPO PLATAFORMA	NÚMERO
PMMER	11
BC10X	1
TOTALES	12

DENOMINACIÓN TÁCTICA	NÚMERO	PESO
CC REC LEOPARDO 2 ER (BÚFALO)	1	63.0
VEC M1	2	16.0
CC LEOPARDO 2 E	8	63.0
TOTALES	11	599.0

COSTE TOTAL (IVA no incluido): 80.140,57 €

RESULTADO TREN 2.

TRAYECTO: Córdoba-El Higuerón -San Gregorio

28/10/2019

COMPOSICIÓN

UIC	TIPO	DENOMINACIÓN TÁCTICA	PESO	C-41	ALTO	LARGO	ANCHO	TIPO DE MATERIAL
507105080034	BC10X							
837139710037	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710052	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710102	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710086	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710383	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710466	PMMER	CC LEOPARDO 2 E	63	X	3.0	9.67	3.75	Carro de combate
837139710532	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710540	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710375	PMMER	VCI/C PIZARRO FASE II	28	null	2.78	7.13	3165.0	Vehículos blindados
837139710482	PMMER	VCI/C PIZARRO PC BON	24	null	2932.0	6836.0	3.15	Vehículos blindados
837139710516	PMMER	VCI/C PIZARRO PC BON	24	null	2932.0	6836.0	3.15	Vehículos blindados
837139710284	PMMER	TOA M-113 A1 MCC TOW	10	null	2.45	4.78	2.64	Vehículos blindados
837139710284	PMMER	TOA M-113 A1 MCC TOW	10	null	2.45	4.78	2.64	Vehículos blindados
837139710169	PMMER	CC M-47 ER3	45	X	3.07	8.4	3.62	Carro de combate

TIPO PLATAFORMA	NÚMERO
PMMER	13
BC10X	1
TOTALES	14

DENOMINACIÓN TÁCTICA	NÚMERO	PESO
VCI/C PIZARRO FASE II	3	28.0
VCI/C PIZARRO PC BON	2	24.0
TOA M-113 A1 MCC TOW	2	10.0
CC LEOPARDO 2 E	6	63.0
CC M-47 ER3	1	45.0
TOTALES	14	575.0

COSTE TOTAL (IVA no incluido): 71.378,38 €

ANEXOS

<u>Apéndice 5.</u> Códigos QR de las respectivas composiciones

Para hacer la respectiva comprobación, se adjuntan en este apéndice los códigos QR asociados a las plataformas y los materiales de las distintas composiciones ferroviarias.

<u>CÓDIGOS QR COMPOSICIÓN TREN 1</u>

PLATAFORMA (UIC)	507105080059	837139710029	837139710201	837139710433	837139710011	837139710144
CONTENIDO	-	CC LEOPARDO 2E	CC LEOPARDO 2E	CC LEOPARDO 2E	CC LEOPARDO 2E	CC LEOPARDO 2E

PLATAFORMA (UIC)	837139710060	837139710276	837139710177	837139710219	837139710136	837139710128
CONTENIDO	CC LEOPARDO 2E	CC LEOPARDO 2E	CC LEOPARDO 2E	VEC M1	VEC M1	CC REC LEOPARDO 2 ER (BÚFALO)

ANEXOS

CÓDIGOS QR COMPOSICIÓN TREN 2

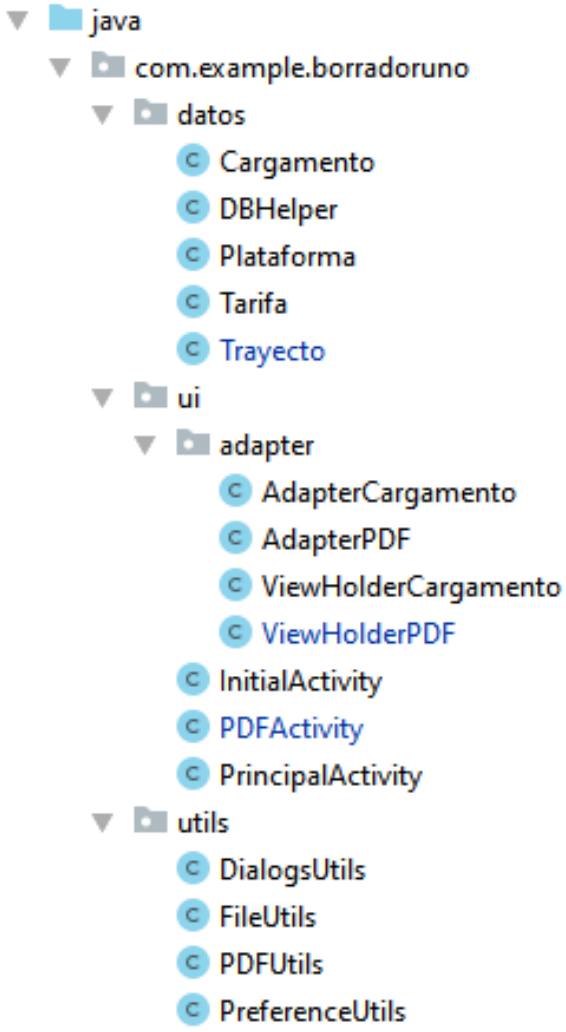
PLATAFORMA (UIC)	507105080034	837139710037	837139710052	837139710102	837139710086	837139710383
CONTENIDO	-	CC LEOPARDO 2E	CC LEOPARDO 2	CC LEOPARDO 2E	CC LEOPARDO 2E	CC LEOPARDO 2E

PLATAFORMA (UIC)	837139710466	837139710532	837139710540	837139710375	837139710482	837139710516
CONTENIDO	CC LEOPARDO 2E	VCI/C PIZARRO FASE II	VCI/C PIZARRO FASE II	VCI/C PIZARRO FASE II	VCI/C PIZARRO PC BON	VCI/C PIZARRO PC BON

PLATAFORMA (UIC)	837139710284	837139710169
CONTENIDO	2X TOA M113 MCCLA TOW	CC M47 ER3

ANEXO F. CÓDIGO FUENTE

En base al proyecto desarrollado, en el presente anexo se recopilan todas las clases de Java empleadas y programadas que han determinado el alcance de este. Dichas clases son las que se reflejan en la siguiente ilustración y, cuya función, en líneas generales, ya se ha definido en el apartado 5 de la memoria. Además, en dicho código se han añadido los comentarios pertinentes para facilitar la legibilidad del mismo.



CÓDIGO FUENTE:

Java Classes

InitialActivity

```
package com.example.borradoruno.ui;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

import com.example.borradoruno.datos.DBHelper;
import com.example.borradoruno.R;

public class InitialActivity extends AppCompatActivity{
    public static DBHelper dbHelper;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        dbHelper = new DBHelper(this);

        Button button =(Button)this.findViewById(R.id.button);

        final View view = getWindow().getDecorView().getRootView();

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view){

                Intent button = new Intent(view.getContext(),
PrincipalActivity.class);
                startActivity(button);
            }
        });

    }
}
```

PrincipalActivity

```
package com.example.borradoruno.ui;

import android.Manifest;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Pair;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.RadioGroup;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.borradoruno.R;
import com.example.borradoruno.datos.Cargamento;
import com.example.borradoruno.datos.Plataforma;
import com.example.borradoruno.datos.Tarifa;
import com.example.borradoruno.datos.Trayecto;
import com.example.borradoruno.ui.adapter.AdapterCargamento;
import com.example.borradoruno.ui.adapter.ViewHolderCargamento;
import com.example.borradoruno.utils.DialogsUtils;
import com.example.borradoruno.utils.FileUtils;
import com.example.borradoruno.utils.PreferenceUtils;
import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionDeniedResponse;
import com.karumi.dexter.listener.PermissionGrantedResponse;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.single.PermissionListener;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;


public class PrincipalActivity extends AppCompatActivity implements
ViewHolderCargamento.CargamentoListener {

    //  Controles
    private Spinner spinnerTrayectos;
    private RadioGroup radioGroupOperacion;
```

```
    private TextView textViewPlataformaActual;
    private RecyclerView recyclerViewCargamento;
    private TextView textViewPlataformaActual_Peso;
    private ImageButton imageButtonRemovePlataforma;

    private List<Plataforma> listPlataformas;  // Todas las
plataformas guardadas
    private Plataforma plataformaActual;  //  Plataforma escaneada
    private boolean pesoTotalSuperiorPermitido;
    private int indexPlataformaActual;

    private ImageButton imageButtonPlataforma_Prev;
    private ImageButton imageButtonPlataforma_Next;
    private TextView textViewPlataformaIndex;

    // Tipo de operación
    private enum TipoOperacion { toSelectPlataforma,
toSelectCargamento }
    private TipoOperacion tipoOperacion;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);
        setTitle("COMPOSICIÓN");

      //  Recoger controles
        spinnerTrayectos = findViewById(R.id.spinnerTrayectos);
        radioGroupOperacion = findViewById(R.id.radioGroupOperacion);
        textViewPlataformaActual  =
findViewById(R.id.textViewPlataformaActual);
        recyclerViewCargamento  =
findViewById(R.id.recyclerViewCargamento);
        textViewPlataformaActual_Peso  =
findViewById(R.id.textViewPlataformaActual_Peso);
        imageButtonPlataforma_Next  =
findViewById(R.id.imageButtonPlataforma_Next);
        imageButtonPlataforma_Prev  =
findViewById(R.id.imageButtonPlataforma_Prev);
        textViewPlataformaIndex  =
findViewById(R.id.textViewPlataformaIndex);
        imageButtonRemovePlataforma  =
findViewById(R.id.imageButtonRemovePlataforma);

        // Gestión del spinner para mostrar trayectos
////////////////////////////////////////////////////
        //  Conseguir la lista de trayectos
        List<Trayecto> listTrayectos  = Trayecto.getListTrayectos();

  //  Convertir la lista de trayectos en una lista de Strings para que
lo entienda el adapter del spinner
        List<String> listNombreTrayectos = new ArrayList<>();
        listNombreTrayectos.add("Seleccione un trayecto");   //*
        for (Trayecto trayecto  : listTrayectos)
            listNombreTrayectos.add(trayecto.getDescription());

   // Crear el adapter con la lista de trayectos en texto, y asociarlo
al spinner
        ArrayAdapter<String> adapterTrayectos = new
ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
listNombreTrayectos);
```

```
        spinnerTrayectos.setAdapter(adapterTrayectos);


////////////////////////////////////////////////////////////////////////
//////////////

        //  Eventos
        radioGroupOperacion.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup radioGroup, int i)
{

                switch (radioGroupOperacion.getCheckedRadioButtonId())
{
                    case R.id.radioButtonPlataforma:    tipoOperacion
= TipoOperacion.toSelectPlataforma; break;
                    case R.id.radioButtonMaterial:      tipoOperacion
= TipoOperacion.toSelectCargamento; break;
                }

            }
        });

    // Estado inicial de la ficha
        recyclerViewCargamento.setAdapter(new
AdapterCargamento(this));
        radioGroupOperacion.check(R.id.radioButtonPlataforma);

   //  Configurar listado de cargamento y crear la plataforma inicial
        recyclerViewCargamento.setLayoutManager(new
LinearLayoutManager(this));

   // Recoger tren desde preferencias si existe
        getTrainFromPreferences();

        indexPlataformaActual = 0;
        plataformaActual = listPlataformas.get(0);
        showPlataformaActual();
    }



    private void getTrainFromPreferences() {

//  Si la lista de plataformas no existe, es que se han perdido al
pasar la app a segundo plano
 //  En ese caso se intenta cargar la información desde preferencias

            int indexPlataforma  = 0;
            String textPlataforma= null;
            do {
                textPlataforma =
PreferenceUtils.GetStringFromDefaultsPrefs(this,
Plataforma.PREF_FIELD_PLATAFORMA + indexPlataforma, null);

                //  Si existe la clave buscada (PLATAFORMA3), tendrá
contenido PLATAFORMA3 453543,345345,345345, El primer número es el uic
de la plataforma y el resto son los ids de los cargamentos

                if (textPlataforma != null) {
```

```
                    String values[]  = textPlataforma.split(",");

                    Plataforma plataforma  =
Plataforma.getPlataforma(values[0]);
                    if (plataforma != null) {

                        for (int i = 1; i < values.length; i++) {
                            Cargamento cargamento =
Cargamento.getCargamento(values[i]);
                            if (cargamento != null)
                                plataforma.addCargamento(cargamento);
                        }

                        if (listPlataformas == null)
                            listPlataformas = new ArrayList<>();

                        listPlataformas.add(plataforma);
                    }
                }

                indexPlataforma++;
            }
            while (textPlataforma != null);

            if (listPlataformas == null) {
                listPlataformas  = new ArrayList<>();
                buttonAddPlataforma_Click(null);
            }
    }

    @Override
    protected void onStop() {
        super.onStop();

        saveAllPlatformsToPrefs();

    }

    /** Este evento se ejecuta al volver a este activity, en este caso
desde la pantalla de captura de Zxing */
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {

        IntentResult result =
IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
        if (result != null   &&   result.getContents() != null) {
                handleQRResult(result.getContents());
        } else {
            super.onActivityResult(requestCode, resultCode, data);
        }
    }


    /** Botón de escanear, se pide o se comprueba el permiso de acceso
a la cámara y si es correcto se llama a Zxing */
    public void btnEscanear(View v){

        // No se puede escanear si no hay plataforma escogida
        if (!plataformaActual.isPlataformaIsValid()   &&
```

```
tipoOperacion  == TipoOperacion.toSelectCargamento) {
            Toast.makeText(this, "Debe escanear primero una
plataforma", Toast.LENGTH_LONG).show();
            return;
        }

        //  Si hay una plataforma y se quiere escoger otra, se
solicita confirmación
        else if (plataformaActual.isPlataformaIsValid()   &&
tipoOperacion  == TipoOperacion.toSelectPlataforma) {

            AlertDialog.Builder builder =new
AlertDialog.Builder(this);
            builder.setTitle("Cambio de plataforma");
            builder.setMessage("Va a cambiar de plataforma, ¿está
seguro?");
            builder.setPositiveButton("Adelante", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface,
int i) {

                    scanQR();
                }
            });
            builder.setNegativeButton("Cancelar", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface,
int i) {

                }
            });
            AlertDialog alertDialog = builder.create();
            alertDialog.show();
        }

        // Se va a escanear un cargamento y hay plataforma, o se va a
escanear la primera plataforma
        else
            scanQR();

    }

    public void buttonAddPlataforma_Click(View view) {

        listPlataformas.add(new Plataforma());
        indexPlataformaActual = listPlataformas.size()-1;
        plataformaActual = listPlataformas.get(indexPlataformaActual);

        showPlataformaActual();

    }

    public void buttonRemovePlataforma_Click(View view) {

        DialogsUtils.ShowDialogActionYesNo(this, "Atención", "Va a
eliminar la plataforma actual, ¿está seguro?", "Eliminar", "Cancelar",
new DialogInterface.OnClickListener() {
            @Override
         public void onClick(DialogInterface dialogInterface, int i) {
```

```
                listPlataformas.remove(plataformaActual);

                // Obtener el índice de la nueva plataforma
                if (indexPlataformaActual > 0)
                    indexPlataformaActual--;

                plataformaActual  =
listPlataformas.get(indexPlataformaActual);
                showPlataformaActual();

                saveAllPlatformsToPrefs();
            }
        });
    }

    public void buttonPlataformaPref_Click(View view) {

        indexPlataformaActual--;
        plataformaActual = listPlataformas.get(indexPlataformaActual);

        showPlataformaActual();
    }

    public void buttonPlataformaNext_Click(View view) {

        indexPlataformaActual++;
        plataformaActual = listPlataformas.get(indexPlataformaActual);

        showPlataformaActual();
    }

 /** Muestra y ajusta todos los controles asociados a la posición de
la plataforma actual */
    public void showPlataformaIndex() {

        int numPlataformas  = listPlataformas.size();
        textViewPlataformaIndex.setText(numPlataformas == 1  ?
"Plataforma 1"  : "Plataforma " + (indexPlataformaActual+1) + " de " +
numPlataformas);
        imageButtonPlataforma_Prev.setVisibility(indexPlataformaActual
> 0  ? View.VISIBLE  : View.INVISIBLE);
        imageButtonPlataforma_Next.setVisibility(indexPlataformaActual
< numPlataformas-1  ? View.VISIBLE  : View.INVISIBLE);
        imageButtonRemovePlataforma.setVisibility(numPlataformas <= 1
? View.GONE  : View.VISIBLE);
    }

    /** Proceder con el escaneo, primero se comprueban los permisos */
    private void scanQR() {

        // Capturar QR comprobando primero el prermiso
        Dexter.withActivity(this)
                .withPermission(Manifest.permission.CAMERA)
                .withListener(new PermissionListener() {
                    @Override
                    public void
onPermissionGranted(PermissionGrantedResponse response) {
                        startQRScanner();
                    }
```

```
                    @Override
                    public void
onPermissionDenied(PermissionDeniedResponse response) {
                        Toast.makeText(PrincipalActivity.this, "No has
dado permiso para usar la cámara", Toast.LENGTH_LONG).show();
                    }

                    @Override
                    public void
onPermissionRationaleShouldBeShown(PermissionRequest permission,
PermissionToken token) {

                        if
(ActivityCompat.shouldShowRequestPermissionRationale(PrincipalActivity
.this, permission.getName()))
                            Toast.makeText(PrincipalActivity.this, "Si
no das permiso no podrás capturar el QR", Toast.LENGTH_LONG).show();

                    }
                }).check();
    }


    /** Acciones a realizar con un QR capturado correctamente */
    public void handleQRResult(String codeScanned) {

        switch (tipoOperacion) {
            case toSelectPlataforma:    Plataforma plataformaScanned =
Plataforma.getPlataforma(codeScanned);
                                        if (plataformaScanned == null)
                                            Toast.makeText(this,
"Código de plataforma inválido", Toast.LENGTH_LONG).show();
                                        else {

plataformaActual.CopyInfo(plataformaScanned);
                                            showPlataformaActual();
                                        }
                                        break;

            case toSelectCargamento:
selectCargamento(Cargamento.getCargamento(codeScanned)); break;
        }

        // Guardar en preferencias
        saveAllPlatformsToPrefs();
    }

    /** Mostrar datos de la plataforma actual */
    private void showPlataformaActual() {

        //  Añadir el listado de cargamentos al recyclerView

((AdapterCargamento)recyclerViewCargamento.getAdapter()).setListCargam
entos(plataformaActual.getListCargamentos());

        // Mostrar sus propiedades

textViewPlataformaActual.setText(plataformaActual.getDescription());

        // Mostrar info de cargamentos (por ahora vacía)
        showCargamentosInfo();
```

```
        showPlataformaIndex();

    }
  /**  Se acaba de escanear un cargamento, se muestra su descripción,
     * y se comprueban los pesos
     *
     * Si cargamento vale null, se ignora*/
    private void selectCargamento(Cargamento cargamento) {

        if (cargamento == null) {
            Toast.makeText(this, "Código de cargamento inválido",
Toast.LENGTH_LONG).show();
            return;
        }

        // Añadir el cargamento a la lista
        plataformaActual.addCargamento(cargamento);

        //  Recargar el listado
        recyclerViewCargamento.getAdapter().notifyDataSetChanged();

        // Mostrar el sumatorio de pesos
        showCargamentosInfo();

    }

/** Muestra número de cargamentos y la suma de peso
     * Se mostrará en rojo si es superior a lo permitido y se mostrará
un aviso */
    private void showCargamentosInfo() {

        if (!plataformaActual.isPlataformaIsValid()) {

textViewPlataformaActual_Peso.setVisibility(View.INVISIBLE);
            return;
        }

        textViewPlataformaActual_Peso.setVisibility(View.VISIBLE);

        int numCargamentos  = plataformaActual.getNumCargamentos();
        String numCargamentosText  = numCargamentos == 0  ? "Ningún
cargamento"  : numCargamentos == 1  ? "1 Cargamento - "  :
numCargamentos + " Cargamentos - ";
        String pesoTotalText  = String.format("%.2f",
plataformaActual.calculateSumPesos()) + " Tm";
        textViewPlataformaActual_Peso.setText(numCargamentosText +
(numCargamentos >0 ? pesoTotalText:""));

        pesoTotalSuperiorPermitido = plataformaActual.pesoExcedido();

textViewPlataformaActual_Peso.setTextColor(pesoTotalSuperiorPermitido
? Color.RED  : Color.DKGRAY);

        if (pesoTotalSuperiorPermitido)
            Toast.makeText(this, "Peso de la plataforma excedido",
Toast.LENGTH_LONG).show();


    }
```

```
    /** Iniciar la captura QR */
    private void startQRScanner() {
        new IntentIntegrator(this).initiateScan();
    }



    ////////////////////////////////////////
    // INTERACCIÓN CON EL LISTADO
    ////////////////////////////////////////
 /** Este método se llamará desde la fila donde se ha pulsado el botón
de borrar */
    @Override
    public void onCargamentoDelete(final Cargamento cargamento) {

        AlertDialog.Builder builder =new AlertDialog.Builder(this);
        builder.setTitle("Eliminar cargamento");
        builder.setMessage("Va a eliminar el cargamento " +
cargamento.getDenominacionTactica() + " ¿está seguro?");
        builder.setPositiveButton("Eliminar", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int
i) {

              // Eliminar el cargamento de la lista
                plataformaActual.removeCargamento(cargamento);

              //recalcular pesos
                showCargamentosInfo();

             //  Actualizar el listado

recyclerViewCargamento.getAdapter().notifyDataSetChanged();

            }
        });
        builder.setNegativeButton("Cancelar", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int
i) {     }
        });
        AlertDialog alertDialog = builder.create();
        alertDialog.show();

    }


    @Override
    public void onCargamentoShowInfo(Cargamento cargamento) {

    }


    /** Botón para generar documento CSV */
    public void   buttonGenerateCSV_Click(View view) {
```

```
        // Primera comprobación: debe haber una ruta seleccionada
        int indexRuta  = spinnerTrayectos.getSelectedItemPosition();
        if (indexRuta == 0) {
            Toast.makeText(this, "Debes escoger una ruta",
Toast.LENGTH_SHORT).show();
            return;
        }

        //  Segunda comprobación: hay plataformas inválidas
        boolean thereIsInvalidPlataforms  =  false;
        for (Plataforma plataforma  : listPlataformas)
            if (!plataforma.isPlataformaIsValid()) {
                thereIsInvalidPlataforms = true;
                break;
            }

        //  Tercera comprobación: hay plataformas con peso excedido
        for (Plataforma plataforma  : listPlataformas)
            if (plataforma.pesoExcedido()) {
                Toast.makeText(this, "Plataforma con peso excedido",
Toast.LENGTH_LONG).show();
                return;
            }

        //  Comprobar si hay alguna inválida
        if (thereIsInvalidPlataforms)
                DialogsUtils.ShowDialogActionYesNo(this, "Atención",
"Hay plataformas que no ha escaneado", "Desecharlas y continuar",
"Cancelar el documento", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface
dialogInterface, int i) {

      //  Si se pulsa el botón de continuar, se genera el csv
                        generateCSV();
                    }
                });
        else
            generateCSV();
    }

/**  Generar el documento csv una vez realizadas las comprobaciones */
    private void generateCSV() {

        //  Cabecera trayecto
        Trayecto trayectoSelected =
Trayecto.getListTrayectos().get(spinnerTrayectos.getSelectedItemPositi
on()-1);  //  El -1 es por la línea añadida 'seleccione trayecto'
        String textCSV  = "TRAYECTO: " + trayectoSelected.getNombre()
+ "\n\n";

        //  Añadir fecha
        SimpleDateFormat simpleDateFormat  = new
SimpleDateFormat("dd/MM/yyyy", Locale.getDefault());
        String date  = simpleDateFormat.format(new Date());
        textCSV  = textCSV + date + "\n\n";

        textCSV  = textCSV + "COMPOSICIÓN\n\n";

        //  Cabecera materiales
        textCSV = textCSV + "UIC;TIPO;DENOMINACIÓN TÁCTICA;PESO;C-
```

```
41;ALTO;LARGO;ANCHO;TIPO DE MATERIAL\n";

        //  Generar lineas materiales
        for (Plataforma plataforma  : listPlataformas)
            if (plataforma.isPlataformaIsValid()) {

                String cabeceraPlataformaCSV = "=\"" +
plataforma.getUic() + "\";" + plataforma.getTipo();

                if (plataforma.hasCargamentos())
                    for (Cargamento cargamento  :
plataforma.getListCargamentos()) {

                        String cargamentoCSV  = cabeceraPlataformaCSV
+ ";" + cargamento.getDenominacionTactica() + ";" +
String.format("%.2f", cargamento.getPeso()) + ";" +
cargamento.getC41() + ";" + cargamento.getAlto() + ";" +
cargamento.getLargo() + ";" + cargamento.getAncho() + ";" +
cargamento.getTipomaterial() + "\n";
                        textCSV  = textCSV + cargamentoCSV;
                    }
                else {
                    textCSV  = textCSV + cabeceraPlataformaCSV + "\n";
                }


            }

//  Tabla resumen plataformas
////////////////////////////////////////////////////////////

        //  Cabecera
        textCSV  = textCSV + "\n\nTIPO PLATAFORMA;NÚMERO\n";

        // Obtener resumen de plataformas y para cada una el número de
veces que se repite
        HashMap<String, Integer> listPlataformasByTipo  = new
HashMap<>();
        for (Plataforma plataforma  : listPlataformas)
            if (plataforma.isPlataformaIsValid()) {

                String key = plataforma.getTipo();

                int numPlataformas  =
listPlataformasByTipo.containsKey(key) ?
listPlataformasByTipo.get(key)  : 0;
                listPlataformasByTipo.put(key, numPlataformas + 1);
            }

        // Mostrar resumen de plataformas
        for (String plataformaTipo  : listPlataformasByTipo.keySet())
            textCSV  = textCSV + plataformaTipo + ";" +
listPlataformasByTipo.get(plataformaTipo) + "\n";

        // Totales
        textCSV  = textCSV + "TOTALES;" + listPlataformas.size() +
"\n";

 // Tabla resumen cargamentos
////////////////////////////////////////////////////
```

```
        //  Cabecera
        textCSV  = textCSV + "\n\nDENOMINACIÓN TÁCTICA;NÚMERO;PESO\n";
// Obtener resumen de plataformas y para cada una el número de veces
que se repite
        HashMap<String, Pair<Cargamento, Integer>>
listCargamentosByTipo  = new HashMap<>();
        for (Plataforma plataforma  : listPlataformas)
            if (plataforma.isPlataformaIsValid())
                for (Cargamento cargamento  :
plataforma.getListCargamentos()) {

                    String key = cargamento.getDenominacionTactica();

                    int numCargamentos  =
listCargamentosByTipo.containsKey(key) ?
listCargamentosByTipo.get(key).second  : 0;
                    listCargamentosByTipo.put(key,
Pair.create(cargamento, numCargamentos + 1));
                }

        // Mostrar resumen de cargamentos
        double totalPesos = 0;
        int numTotalCargamentos = 0;
        for (String cargamentoDenominación  :
listCargamentosByTipo.keySet()) {

            Cargamento cargamento =
listCargamentosByTipo.get(cargamentoDenominación).first;
            int numCargamento =
listCargamentosByTipo.get(cargamentoDenominación).second;

            textCSV  = textCSV + cargamentoDenominación + ";" +
numCargamento + ";" + cargamento.getPeso() + "\n";
            totalPesos += cargamento.getPeso() * numCargamento;

            numTotalCargamentos += numCargamento;
        }

        // Totales
        textCSV  = textCSV + "TOTALES;" + numTotalCargamentos + ";" +
totalPesos + "\n\n";

        // Mostrar tasación del tren (la suma es sin iva)
        textCSV = textCSV + " COSTE TOTAL (IVA no incluido):;" +
String.format("%.2f ", Tarifa.CalcularTarifa(listPlataformas,
trayectoSelected.getDistanciaKms())) + "€;";

        //  Generar fichero
        SimpleDateFormat simpleDateFormatForFilename  = new
SimpleDateFormat("dd-MM-yyyy", Locale.getDefault());
        String dateForFilename  =
simpleDateFormatForFilename.format(new Date());
        String filename  = trayectoSelected.getNombre() + "_" +
dateForFilename + ".csv";

        FileUtils.WriteFile(this, filename, textCSV);

        DialogsUtils.ShowDialogActionOk(this, "Atención", "Se ha
generado un informe de con el nombre '" + filename + "' en la carpeta
de descargas.\nCuando se asegure que es correcto y no va a realizar
ninguna modificación en la información de las plataformas o materiles,
```

```
pulse en el menú la opción 'Cerrar tren'.", "Entendido");
    }


    ////////////////////
    // MENÚ SUPERIOR
    ////////////////////


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_principal, menu);

        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {

        switch (item.getItemId()) {
            case R.id.action_close_train: actionCloseTrain(); break;
            case R.id.action_info_pdf: actionInfoPdf(); break;
        }


        return super.onOptionsItemSelected(item);
    }

    private void actionCloseTrain() {

        DialogsUtils.ShowDialogActionYesNo(this, "Atención", "Va a
cerrar este tren.\n\nSi confirma la operación se eliminará toda la
información asociada.\n\nAsegurese que ha generado el fichero 'csv',
lo tiene en lugar seguro, y es correcto.", "Cerrar tren", "Cancelar",
new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int
i) {


DialogsUtils.ShowDialogActionYesNo(PrincipalActivity.this, "Atención",
"¿Está seguro?", "Cerrar tren", "Cancelar", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface
dialogInterface, int i) {

         // Eliminar todas las plataformas de preferencias
                        removeAllPlatformsFromPrefs();

                        // Borrar el tren de memoria
                        listPlataformas.clear();

                        //  Reiniciar ficha

                        //  Reinciar spinner
                        spinnerTrayectos.setSelection(0);

   //  Crear la plataforma inicial (aunque esté vacía) y mostrarla
      //  Esta llamada mostrará la plataforma recién creada
```

```
                        buttonAddPlataforma_Click(null);
                    }
                });

            }
        });
    }
    private void removeAllPlatformsFromPrefs() {

        int index = 0;
        String plataforma = null;
        do {

            String plataformaPrefKey =
Plataforma.PREF_FIELD_PLATAFORMA + index;
            plataforma =
PreferenceUtils.GetStringFromDefaultsPrefs(PrincipalActivity.this,
plataformaPrefKey, null);

            if (plataforma != null)

PreferenceUtils.RemovePrefInDefaultPrefs(PrincipalActivity.this,
plataformaPrefKey);

            index++;
        }
        while (plataforma != null);
    }
    private void saveAllPlatformsToPrefs() {

        removeAllPlatformsFromPrefs();

        int index = 0;
        for (Plataforma plataforma : listPlataformas)
            if (plataforma.isPlataformaIsValid())
                plataforma.savePlataformaInPreferencias(this,
index++);
    }

    private void actionInfoPdf() {

        startActivity(new Intent(this, PDFActivity.class));
    }
}
```

PDFActivity

```
package com.example.borradoruno.ui;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.Manifest;
import android.os.Bundle;
import android.view.MenuItem;

import com.example.borradoruno.R;
import com.example.borradoruno.ui.adapter.AdapterPDF;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionDeniedResponse;
import com.karumi.dexter.listener.PermissionGrantedResponse;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.single.BasePermissionListener;
import com.karumi.dexter.listener.single.PermissionListener;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class PDFActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pdf);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);  //
Mostrar flecha de volver
        setTitle("Ficheros de ayuda");

        // Obtener listado de pdfs
        List<String> listPDFs = new ArrayList<>();
        try {

            String listFilesInAssets[] = getAssets().list("pdf");
            listPDFs.addAll(Arrays.asList(listFilesInAssets));  // Si
se usa directamente listPDFs = Arrays.asList(listFilesInAssets), se
crea una lista de tamaño fijo, pero es necesario eliminar el fichero
de la bd



        } catch (IOException e) {
            e.printStackTrace();
        }

        RecyclerView recyclerViewPdfs  =
findViewById(R.id.recyclerViewPdfs);
        recyclerViewPdfs.setLayoutManager(new
LinearLayoutManager(this));
        recyclerViewPdfs.setAdapter(new AdapterPDF(listPDFs));


        Dexter.withActivity(this)
```

```
.withPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE)
                .withListener(new PermissionListener() {
                    @Override
                    public void
onPermissionGranted(PermissionGrantedResponse response) {

                    }

                    @Override
                    public void
onPermissionDenied(PermissionDeniedResponse response) {

                    }

                    @Override
                    public void
onPermissionRationaleShouldBeShown(PermissionRequest permission,
PermissionToken token) {

                    }
                }).check();

    }


    /** Para que la flecha de volver funcione, debe añadirse este
método de control del menú */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        if (item.getItemId() == android.R.id.home) {
            finish();
        }

        return super.onOptionsItemSelected(item);

    }
}
```

DBHelper

```
package com.example.borradoruno.datos;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.text.TextUtils;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

public class DBHelper extends SQLiteOpenHelper {
    // Si se cambia la estructura de la base de datos se debe
incrementar el número de versión
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "plataforma.db";
    public final SQLiteDatabase database;

    private boolean dbOk;


    public DBHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);

        // Copiamos la base de datos como asset
        try {
            OutputStream myOutput = new
FileOutputStream(context.getApplicationInfo().dataDir + "/" +
DATABASE_NAME);
            byte[] buffer = new byte[1024];
            int length;
            InputStream myInput =
context.getAssets().open(DATABASE_NAME);
            while ((length = myInput.read(buffer)) > 0) {
                myOutput.write(buffer, 0, length);
            }
            myInput.close();
            myOutput.flush();
            myOutput.close();
        } catch (Exception ex) {
            dbOk = false;
        } finally {
            dbOk = true;
            database =
SQLiteDatabase.openDatabase(context.getApplicationInfo().dataDir + "/"
+ DATABASE_NAME, null, SQLiteDatabase.OPEN_READONLY);
        }
    }

    public boolean getDbOk()
    {
        return dbOk;
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

    }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int
i1) {

    }

    public Cursor getTableData(String table) {
        return database.rawQuery("select * from " + table, null);
    }

    public Cursor getTableData(String[] fields, String[] tables,
String where, String[] values) {
        if (fields == null)
            throw new IllegalArgumentException("Debe indicarse el
parámetro fields");

        if (tables == null)
            throw new IllegalArgumentException("Debe indicarse el
parámetro tables");

        String _fields = TextUtils.join(",", fields);
        String _tables = TextUtils.join(",", tables);
        String sql = "select " + _fields + " from " + _tables +
((where != null) ? " where " + where : "");

        return database.rawQuery(sql, values);
    }
}
```

Cargamento

```
package com.example.borradoruno.datos;

import android.database.Cursor;

import com.example.borradoruno.ui.InitialActivity;

public class Cargamento {
    private String denominacionTactica;
    private String noc;
    private double  alto;
    private double ancho;
    private double largo;
    private String tipomaterial;
    private String c41;
    private double peso;

    public Cargamento (String denominacionTactica,String noc, double
alto, double ancho, double largo,
                       String tipomaterial, String c41, double peso){
        this.denominacionTactica=denominacionTactica;
        this.noc=noc;
        this.alto=alto;
        this.ancho=ancho;
        this.largo=largo;
        this.tipomaterial=tipomaterial;
        this.peso=peso;
        this.c41=c41;

    }

    public String   getDenominacionTactica(){
        return denominacionTactica;
    }
    public String getNoc() {
        return noc;
    }
    public double getAlto() {
        return alto;
    }
    public double getAncho() {
        return ancho;
    }
    public double getLargo(){
        return largo;
    }
    public String getTipomaterial(){
        return tipomaterial;
    }
    public String getC41(){
        return c41;
    }
     public double getPeso(){
        return peso;
     }

     public static Cargamento   getCargamento(String noc){
         Cursor cursor= InitialActivity.dbHelper.getTableData(new
String[]{"denominaciontactica","noc","alto","ancho",
                 "largo","tipomaterial","c41","peso"},
```

```
                    new String[] {"cargamento"},
                    "noc=?", new String[] {noc});
            if (cursor.getCount()==0)
                    return null;
            else {
                cursor.moveToNext();
                return new Cargamento(cursor.getString(0),
cursor.getString(1),cursor.getDouble(2),cursor.getDouble(3),

cursor.getDouble(4),cursor.getString(5),cursor.getString(6),cursor.get
Double(7));
            }
      }
}
```

Plataforma

```
package com.example.borradoruno.datos;

import android.content.Context;
import android.database.Cursor;

import com.example.borradoruno.ui.InitialActivity;
import com.example.borradoruno.utils.PreferenceUtils;

import java.util.ArrayList;
import java.util.List;

public class  Plataforma {

    private String  uic;
    private String tipo;
    private String propietario;
    private double pesoMax;

    private List<Cargamento> listCargamentos;  // Cargamentos para
ersta plataforma

    public static final String PREF_FIELD_PLATAFORMA = "PLATAFORMA";

    private boolean plataformaIsValid;  // Una plataforma es válida si
se ha escaneado

    /** Constructor para una plataforma vacía sin escanear (no válida)
*/
    public Plataforma() {

        plataformaIsValid = false;

        listCargamentos  = new ArrayList<>();

    }

    /** Constructor para una plataforma recogida de BD */
    public Plataforma(String uic, String tipo, String propietario,
double pesoMax){
        this.uic=uic;
        this.tipo=tipo;
        this.propietario=propietario;
        this.pesoMax=pesoMax;

        listCargamentos  = new ArrayList<>();

        plataformaIsValid = true;
    }

    /** Recoge info de una plataforma escaneada y la coloca en esta
plataforma
     * Esta plataforma pasa a ser válida     */
    public void CopyInfo(Plataforma plataforma) {
        this.uic = plataforma.uic;
        this.tipo = plataforma.tipo;
        this.propietario = plataforma.propietario;
        this.pesoMax = plataforma.pesoMax;

        this.plataformaIsValid = true;
    }
```

```
    public String getUic(){
            return uic;
        }
    public String getTipo(){
        return tipo;
    }
    public String getPropietario(){
        return propietario;
    }
    public double getPesoMax(){
        return pesoMax;
    }


    public boolean isPlataformaIsValid() {        return
plataformaIsValid;    }

    public List<Cargamento> getListCargamentos() { return
listCargamentos; }
    public void addCargamento(Cargamento cargamento)  {
listCargamentos.add(cargamento);    }
    public int getNumCargamentos()  { return listCargamentos.size(); }
    public void removeCargamento(Cargamento cargamento) {
listCargamentos.remove(cargamento); }
    public boolean hasCargamentos() { return
!listCargamentos.isEmpty(); }
    public boolean isEmpty() { return listCargamentos.isEmpty(); }

    public boolean pesoExcedido() {

        return getPesoMax() < calculateSumPesos();
    }

    public float calculateSumPesos() {

        float pesoTotal  = 0;
        for (Cargamento cargamento : listCargamentos)
            pesoTotal += cargamento.getPeso();

        return pesoTotal;
    }

    public String getDescription() { return !plataformaIsValid ? "Debe
escanear una plataforma" :  uic + " - " + tipo; }

    public static Plataforma getPlataforma(String uic)
    {
            Cursor cursor = InitialActivity.dbHelper.getTableData(new
String[]{"uic", "tipo", "propietario", "pesomax"},
                new String[]{"plataformas"},
                "uic = ?",
                new String[]{uic});

        if (cursor.getCount() == 0)
                return null;
        else
        {
            cursor.moveToNext();
            return new Plataforma(cursor.getString(0),
```

```
cursor.getString(1),cursor.getString(2), cursor.getDouble(3));
        }
    }


    public void savePlataformaInPreferencias(Context context, int
indexPlataforma) {

        // PLATAFORMA0  58747848, 67548, 4359458, 482343  (la primera
es 0)
        // PLATAFORMA1  58747848, 67548, 4359458

        // Guardar la plataforma actual
        String textPlataforma  = uic + ",";
        for (Cargamento cargamento  : listCargamentos)
            textPlataforma = textPlataforma + cargamento.getNoc() +
",";

        // Guardar el texto de la plataforma
            PreferenceUtils.PutStringInDefaultPrefs(context,
PREF_FIELD_PLATAFORMA+indexPlataforma, textPlataforma);

    }
}
```

Trayecto

```
package com.example.borradoruno.datos;

import android.database.Cursor;

import com.example.borradoruno.ui.InitialActivity;

import java.util.ArrayList;
import java.util.List;

public class Trayecto {

    private String nombre;
    private int distanciaKms;

    public Trayecto(String nombre, int distanciaKms) {
        this.nombre = nombre;
        this.distanciaKms = distanciaKms;
    }

    public String getNombre() {
        return nombre;
    }

    public int getDistanciaKms() {
        return distanciaKms;
    }

    public String getDescription() {
        return nombre + " (" + distanciaKms + " kms.)";
    }

    public static List<Trayecto> getListTrayectos() {

        Cursor cursor =
InitialActivity.dbHelper.getTableData("TRAYECTOS");
        if (cursor.getCount()==0)
            return null;
        else {

            List<Trayecto> listTrayectos = new ArrayList<>();

            while (cursor.moveToNext()) {
                listTrayectos.add(new Trayecto(cursor.getString(0),
cursor.getInt(1)));
            }

            return listTrayectos;
        }

    }
}
```

Tarifa

```
package com.example.borradoruno.datos;

import android.database.Cursor;

import com.example.borradoruno.ui.InitialActivity;

import java.util.List;

public class Tarifa {
    private String propietario;
    private String tipoMaterial;
    private double tarifa;



    public Tarifa (String propietario, String tipomaterial, double
tarifa){
        this.propietario=propietario;
        this.tipoMaterial=tipomaterial;
        this.tarifa=tarifa;

    }

    public String getPropietario(){
        return propietario;
    }
    public String getTipomaterial() {
        return tipoMaterial;
    }
    public double getTarifa() {
        return tarifa;
    }


    public static Tarifa getTarifa(String propietario, String tipo,
String peso){
        Cursor cursor= InitialActivity.dbHelper.getTableData(new
String[]{"propietario","tipomaterial","peso","tarifa"},
                new String[] {"tarifas"},
                "propietario=?   AND   tipomaterial=?   AND   peso=?",
new String[] {propietario, tipo, peso});
        if (cursor.getCount()==0)
            return null;
        else {
            cursor.moveToNext();
            return new Tarifa(cursor.getString(0),
cursor.getString(1), cursor.getDouble(3));
        }
    }

    private static final String TIPO_BLINDADO  = "Vehículos
blindados";
    private static final String TIPO_OTROS  = "Resto de material";
    private static final String TIPO_VIAJEROS  = "Coche de viajeros";
    private static final String PROPIETARIO_VIAJEROS  = "viajeros";

    public static float CalcularTarifa(List<Plataforma>
listPlataformas, int kms) {
```

```
        float total  = 0f;
        for (Plataforma plataforma  : listPlataformas) {

            // Si una plataforma es de tipo viajeros no se recorren
sus cargamentos y se suma su tarifa directamente
            if
(plataforma.getPropietario().equalsIgnoreCase(PROPIETARIO_VIAJEROS)) {
                Tarifa tarifa =
Tarifa.getTarifa(plataforma.getPropietario(), TIPO_VIAJEROS, "-");

                total += tarifa.getTarifa();
            }
            else
                for (Cargamento cargamento :
plataforma.getListCargamentos()) {

                    //  Valores posibles para peso:
                    // ">=15" o "<15" o "-"
                    // Hay que construir la cadena para encontrar el
precio por el peso apropiado
                    String pesoText = "-";
                    String cargamentoTipo =
cargamento.getTipomaterial();

                    //  Si es blindado el texto del peso depende del
peso del cargamento
                    //  En el resto de casos se usa "-"
                    if
(cargamentoTipo.equalsIgnoreCase(TIPO_BLINDADO))
                        if (cargamento.getPeso() >= 15f)
                            pesoText = ">=15";
                        else
                            pesoText = "<15";

                    // Texto para el tipo de material
                    String cargamentoTipoText = cargamentoTipo;
                    if
(!cargamentoTipo.equalsIgnoreCase(TIPO_BLINDADO))
                        cargamentoTipoText = TIPO_OTROS;

                    // Obtener la tarifa y el precio correspondiente
                    Tarifa tarifa =
Tarifa.getTarifa(plataforma.getPropietario(),
                            cargamentoTipoText,
                            pesoText);

                    double precio = tarifa.getTarifa();

                    // REalizar el cálculo del total
                    total += precio * cargamento.getPeso() * kms;

                }
        }

        return total;
    }
}
```

DialogUtils

```
package com.example.borradoruno.utils;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;

import androidx.appcompat.app.AppCompatActivity;

public class DialogsUtils {

    ////////////////////////////////
    // DIALOGO NORMAL DOS BOTONES
    ////////////////////////////////

    /** Mostrar diálogo de confirmación para realizar alguna acción con dos botones
*/
    public static void ShowDialogActionYesNo(Context context, String title, String
message, String textButtonOk, String textButtonCancel,
DialogInterface.OnClickListener onClickListener) {
        ShowDialogActionYesNo(context, title, message, textButtonOk,
textButtonCancel, onClickListener, null);
    }

    /** Mostrar diálogo de confirmación para realizar alguna acción con dos botones
     * VERSIÓN ID RECURSOS */
    public static void ShowDialogActionYesNo(Context context, int title, int
message, int textButtonOk, int textButtonCancel, DialogInterface.OnClickListener
onClickListener) {
        ShowDialogActionYesNo(context, title, message, textButtonOk,
textButtonCancel, onClickListener, null);
    }

    /** Mostrar diálogo de confirmación para realizar alguna acción con dos botones
*/
    public static void ShowDialogActionYesNo(Context context, String title, String
message, String textButtonOk, String textButtonCancel,
DialogInterface.OnClickListener onClickListenerOk, Dialog.OnClickListener
onClickListenerCancel) {
        ShowDialogActionYesNo(context, title, message, textButtonOk,
textButtonCancel, onClickListenerOk, onClickListenerCancel, null);
    }

    /** Mostrar diálogo de confirmación para realizar alguna acción con dos botones
     * VERSIÓN ID RECURSOS */
    public static void ShowDialogActionYesNo(Context context, int title, int
message, int textButtonOk, int textButtonCancel, DialogInterface.OnClickListener
onClickListenerOk, Dialog.OnClickListener onClickListenerCancel) {
        ShowDialogActionYesNo(context, title, message, textButtonOk,
textButtonCancel, onClickListenerOk, onClickListenerCancel, null);
    }


    /** Mostrar diálogo de confirmación para realizar alguna acción con dos botones
*/
    public static void ShowDialogActionYesNo(Context context, String title, String
message, String textButtonOk, String textButtonCancel,
DialogInterface.OnClickListener onClickListenerOk, Dialog.OnClickListener
onClickListenerCancel, Dialog.OnDismissListener onDismissListener) {
        ShowDialogActionYesNoNeutral(context, title, message, textButtonOk,
        textButtonCancel, null, onClickListenerOk, onClickListenerCancel,
        null, onDismissListener);
    }
```

```
    /** Mostrar diálogo de confirmación para realizar alguna acción con dos botones
     * VERSIÓN ID RECURSOS */
    public static void ShowDialogActionYesNo(Context context, int title, int
message, int textButtonOk, int textButtonCancel, DialogInterface.OnClickListener
onClickListenerOk, Dialog.OnClickListener onClickListenerCancel,
Dialog.OnDismissListener onDismissListener) {
        ShowDialogActionYesNoNeutral(context, title, message, textButtonOk,
textButtonCancel, -1, onClickListenerOk, onClickListenerCancel, null,
onDismissListener);
    }

    /** Mostrar diálogo de confirmación para realizar alguna acción con tres
botones */
    public static void ShowDialogActionYesNoNeutral(Context context, String title,
String message, String textButtonOk, String textButtonCancel, String
textButtonNeutral, DialogInterface.OnClickListener onClickListenerOk,
Dialog.OnClickListener onClickListenerCancel, Dialog.OnClickListener
onClickListenerNeutral, Dialog.OnDismissListener onDismissListener) {

        if (context == null   ||   (context instanceof AppCompatActivity &&
((AppCompatActivity) context).isFinishing()))
            return;

        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        if (title != null)
            builder.setTitle(title);
        if (message != null)
            builder.setMessage(message);
        if (textButtonOk != null)
            builder.setPositiveButton(textButtonOk, onClickListenerOk);
        if (textButtonCancel != null)
            builder.setNegativeButton(textButtonCancel, onClickListenerCancel);
        if (textButtonNeutral != null)
            builder.setNeutralButton(textButtonNeutral, onClickListenerNeutral);
        if (onDismissListener != null)
            builder.setOnDismissListener(onDismissListener);
        AlertDialog dialog = builder.create();
        dialog.show();
    }

    /** Mostrar diálogo de confirmación para realizar alguna acción con tres
botones
     * VERSIÓN ID RECURSOS */
    public static void ShowDialogActionYesNoNeutral(Context context, int title, int
message, int textButtonOk, int textButtonCancel, int textButtonNeutral,
DialogInterface.OnClickListener onClickListenerOk, Dialog.OnClickListener
onClickListenerCancel, Dialog.OnClickListener onClickListenerNeutral,
Dialog.OnDismissListener onDismissListener) {

        if (context == null   ||   (context instanceof AppCompatActivity &&
((AppCompatActivity) context).isFinishing()))
            return;

        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        if (title != -1)
            builder.setTitle(title);
        if (message != -1)
```

```
            builder.setMessage(message);
        if (textButtonOk !=  -1)
            builder.setPositiveButton(textButtonOk, onClickListenerOk);
        if (textButtonCancel != -1)
            builder.setNegativeButton(textButtonCancel, onClickListenerCancel);
        if (textButtonNeutral !=  -1)
            builder.setNeutralButton(textButtonNeutral, onClickListenerNeutral);
        if (onDismissListener != null)
            builder.setOnDismissListener(onDismissListener);
        AlertDialog dialog = builder.create();
        dialog.show();
    }


    ////////////////////////////////
    // DIALOGO NORMAL UN BOTÓN
    ////////////////////////////////

    /** Mostrar diálogo de confirmación con un botón */
    public static void  ShowDialogActionOk(Context context, String title, String
message, String textButtonOk) {
        ShowDialogActionYesNo(context, title, message, textButtonOk, null, null);
    }

    /** Mostrar diálogo de confirmación con un botón
     * VERSIÓN ID RECURSOS */
    public static void ShowDialogActionOk(Context context, int title, int message,
int textButtonOk) {
        ShowDialogActionYesNo(context, title, message, textButtonOk, -1, null);
    }

    /** Mostrar diálogo de confirmación con un botón */
    public static void ShowDialogActionOk(Context context, String title, String
message, String textButtonOk, Dialog.OnDismissListener onDismissListener) {
        ShowDialogActionYesNo(context, title, message, textButtonOk, null, null,
null, onDismissListener);
    }

    /** Mostrar diálogo de confirmación con un botón
     * VERSIÓN ID RECURSOS */
    public static void ShowDialogActionOk(Context context, int title, int message,
int textButtonOk, Dialog.OnDismissListener onDismissListener) {
        ShowDialogActionYesNo(context, title, message, textButtonOk, -1, null,
null, onDismissListener);
    }

}
```

FileUtils

```
package com.example.borradoruno.utils;

import android.Manifest;
import  android.app.Activity;
import android.app.DownloadManager;
import android.os.Environment;
import android.widget.Toast;

import com.karumi.dexter.Dexter;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionDeniedResponse;
import com.karumi.dexter.listener.PermissionGrantedResponse;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.single.PermissionListener;

import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.nio.charset.StandardCharsets;

import static android.content.Context.DOWNLOAD_SERVICE;

public class FileUtils {

    public static void WriteFile(final Activity activity, final String
filename, final String text) {

        Dexter.withActivity(activity)

.withPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE)
                .withListener(new PermissionListener() {
                    @Override
                    public void
onPermissionGranted(PermissionGrantedResponse response) {

                        FileUtils.saveFile(activity, filename, text);
                    }

                    @Override
                    public void
onPermissionDenied(PermissionDeniedResponse response) {

                    }

                    @Override
                    public void
onPermissionRationaleShouldBeShown(PermissionRequest permission,
PermissionToken token) {

                    }
                }).check();


    }

    private static void saveFile(Activity activity, String filename,
String text) {
```

```
        try {
            File root = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTO
RY_DOWNLOADS), "Cuadros de carga");
            if (!root.exists()) {
                root.mkdirs();
            }
            File file = new File(root, filename);
            //FileWriter writer = new FileWriter(file);



            FileOutputStream fileStream = new FileOutputStream(file);
            OutputStreamWriter writer = new
OutputStreamWriter(fileStream, StandardCharsets.UTF_16LE);



            writer.append(text);
            writer.flush();
            writer.close();

            //  Colocar en carpeta de descarga
            DownloadManager downloadManager = (DownloadManager)
activity.getSystemService(DOWNLOAD_SERVICE);
            downloadManager.addCompletedDownload(file.getName(),
file.getName(), true,
"text/plain",file.getAbsolutePath(),file.length(),true);


            Toast.makeText(activity, "Saved",
Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

CÓDIGO FUENTE:

PFDUtils

```
package com.example.borradoruno.utils;

import android.app.DownloadManager;
import android.content.Context;
import android.content.Intent;
import android.content.res.AssetManager;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Environment;
import android.util.Log;

import androidx.core.content.FileProvider;

import com.example.borradoruno.BuildConfig;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.ref.WeakReference;

import static android.content.Context.DOWNLOAD_SERVICE;

/**
 *
 *   Esta clase usa un asynctask para copiar un fichero desde assets a la
carpeta pública de la app, y luego le pide a Android que abra el pdf
 *   La parte de copiar el fichero se ejecuta en otro hilo, la apertura del
fichero se hace tras la copia
 */
public class PDFUtils {
    private static String TAG = PDFUtils.class.getSimpleName();

    private WeakReference<Context> contextWeakReference;
    private String fileNameIn;
    private String fileNameOut;

    public PDFUtils(Context context, String fileNameIn, String filenameOut) {
        this.contextWeakReference = new WeakReference<>(context);
        this.fileNameIn = fileNameIn;//fileNameIn.endsWith("pdf") ? fileNameIn :
fileNameIn + ".pdf";
        this.fileNameOut = filenameOut;
    }

    public void execute() {

        Context context = contextWeakReference.get();
        if (context != null) {
            new CopyFileAsyncTask().execute();
        }
    }


    private class CopyFileAsyncTask extends AsyncTask<Void, Void, File>{



        final String appDirectoryName = BuildConfig.APPLICATION_ID;
        final File fileRoot = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMEN
TS), appDirectoryName);
```

```
        /** El contenido de este método de AsyncTask es el único que se ejecuta
en otro hilo
         * Cuando se ha acabado la tarea, se llama a onPostExecute() * */
        @Override
        protected File doInBackground(Void... params) {

            Context context = contextWeakReference.get();

            AssetManager assetManager = context.getAssets();

            File fileOut = new File(fileRoot, fileNameOut);

            InputStream in = null;
            OutputStream out = null;
            try {
                fileOut.mkdirs();

                if (fileOut.exists()) {
                    fileOut.delete();
                }
                fileOut.createNewFile();

                in = assetManager.open(fileNameIn);
                Log.d(TAG, "In");

                out = new FileOutputStream(fileOut);
                Log.d(TAG, "Out");

                Log.d(TAG, "Copy file");
                copyFile(in, out);

                Log.d(TAG, "Close");
                in.close();

                out.flush();
                out.close();

                return fileOut;

            } catch (Exception e)
            {
                Log.e(TAG, e.getMessage());
            }

            return null;
        }

        private void copyFile(InputStream in, OutputStream out) throws
IOException
        {
            byte[] buffer = new byte[1024];
            int read;
            while ((read = in.read(buffer)) != -1)
            {
                out.write(buffer, 0, read);
            }
        }

        /** Tras copiar el fichero, se pide a Android que abra el pdf */
        @Override
        protected void onPostExecute(File file) {
            super.onPostExecute(file);

            Context context = contextWeakReference.get();
```

CÓDIGO FUENTE:

```
            Uri uri = FileProvider.getUriForFile(context,
BuildConfig.APPLICATION_ID + ".provider", file);

            Intent intent = new Intent(Intent.ACTION_VIEW);
            intent.setDataAndType(
                    uri,
                    "application/pdf");
            intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

            context.startActivity(intent);

        }
    }
}
```

CÓDIGO FUENTE:

PreferenceUtils

```
package com.example.borradoruno.utils;

import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;

public class    PreferenceUtils {


    public static String GetStringFromDefaultsPrefs(Context context,
String campo, String defaultValue) {
        return
PreferenceManager.getDefaultSharedPreferences(context).getString(campo
, defaultValue);
    }

    public static int GetIntegerFromDefaultsPrefs(Context context,
String campo, int defaultValue) {
        return
PreferenceManager.getDefaultSharedPreferences(context).getInt(campo,
defaultValue);
    }

    public static boolean GetBooleanFromDefaultsPrefs(Context context,
String campo, boolean defaultValue) {
        return
PreferenceManager.getDefaultSharedPreferences(context).getBoolean(camp
o, defaultValue);
    }




    public static void PutStringInDefaultPrefs(Context context, String
campo, String value) {
        SharedPreferences.Editor editorPrefs =
PreferenceManager.getDefaultSharedPreferences(context).edit();
        editorPrefs.putString(campo, value);
        editorPrefs.apply();
    }

    public static void PutBooleanInDefaultPrefs(Context context,
String campo, boolean value) {
        SharedPreferences.Editor editorPrefs =
PreferenceManager.getDefaultSharedPreferences(context).edit();
        editorPrefs.putBoolean(campo, value);
        editorPrefs.apply();
    }


    public static void PutIntegerInDefaultPrefs(Context context,
String campo, int value) {
        SharedPreferences.Editor editorPrefs =
PreferenceManager.getDefaultSharedPreferences(context).edit();
        editorPrefs.putInt(campo, value);
        editorPrefs.apply();
    }

    public static void RemovePrefInDefaultPrefs(Context context,
```

CÓDIGO FUENTE:

```
String campo) {
        SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
        editor.remove(campo);
        editor.apply();
    }


    public static boolean GetBoolean(Context context, String
prefsFile, String campo, boolean defaultValue) {
        return context.getSharedPreferences(prefsFile,
Context.MODE_PRIVATE).getBoolean(campo, defaultValue);
    }

    public static String GetString(Context context, String prefsFile,
String campo, String defaultValue) {
        return context.getSharedPreferences(prefsFile,
Context.MODE_PRIVATE).getString(campo, defaultValue);
    }

    public static int GetInteger(Context context, String prefsFile,
String campo, int defaultValue) {
        return context.getSharedPreferences(prefsFile,
Context.MODE_PRIVATE).getInt(campo, defaultValue);
    }

    public static float GetFloat(Context context, String prefsFile,
String campo, float defaultValue) {
        return context.getSharedPreferences(prefsFile,
Context.MODE_PRIVATE).getFloat(campo, defaultValue);
    }




    public static void PutBoolean(Context context, String prefsFile,
String campo, boolean value) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(prefsFile, Context.MODE_PRIVATE).edit();
        editor.putBoolean(campo, value);
        editor.apply();
    }

    public static void PutString(Context context, String prefsFile,
String campo, String value) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(prefsFile, Context.MODE_PRIVATE).edit();
        editor.putString(campo, value);
        editor.apply();
    }

    public static void PutInteger(Context context, String prefsFile,
String campo, int value) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(prefsFile, Context.MODE_PRIVATE).edit();
        editor.putInt(campo, value);
        editor.apply();
    }
```

CÓDIGO FUENTE:

```
    public static void PutFloat(Context context, String prefsFile,
String campo, float value) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(prefsFile, Context.MODE_PRIVATE).edit();
        editor.putFloat(campo, value);
        editor.apply();
    }


}
```

CÓDIGO FUENTE:

AdapterCargamento

```
package com.example.borradoruno.ui.adapter;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.borradoruno.datos.Cargamento;
import com.example.borradoruno.R;

import java.util.List;

public class AdapterCargamento extends
RecyclerView.Adapter<ViewHolderCargamento> {

    // LIstado de cargamentos, que llega cuando se crea una nueva plataforma
tras escanear QR (método setListCargamentos()  )
    private List<Cargamento> listCargamentos;

    //  Listener que atiende a los eventos de la fila
    private ViewHolderCargamento.CargamentoListener cargamentoListener;

    /** Constructor del adapter     */
    public AdapterCargamento(ViewHolderCargamento.CargamentoListener
cargamentoListener) {
        this.cargamentoListener = cargamentoListener;
    }

    /** Este método se llama cada vez que se necesita crear una nueva fila*/
    @NonNull
    @Override
    public ViewHolderCargamento onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {

        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_cargamento,
parent, false);
        return new ViewHolderCargamento(v, cargamentoListener);
    }

    /** Este método se llama cada vez que se va a pintar un cargamento en una
fila ya creada */
    @Override
    public void onBindViewHolder(@NonNull ViewHolderCargamento holder, int
position) {
        holder.showCargamento(listCargamentos.get(position));
    }

    @Override
    public int getItemCount() {
        return listCargamentos == null  ? 0 : listCargamentos.size();
    }

    public void setListCargamentos(List<Cargamento> listCargamentos) {
        this.listCargamentos = listCargamentos;
        notifyDataSetChanged();
    }

}
```

AdapterPDF

```
package com.example.borradoruno.ui.adapter;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.borradoruno.R;

import java.util.List;

public class AdapterPDF extends RecyclerView.Adapter<ViewHolderPDF> {

    private List<String> listPDFs;

    public AdapterPDF(List<String> listPDFs) {
        this.listPDFs = listPDFs;
    }

    @NonNull
    @Override
    public ViewHolderPDF onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {

        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_pdf,
parent, false);
        return new ViewHolderPDF(v, parent.getContext());
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolderPDF holder, int
position) {
        holder.showPdf(listPDFs.get(position));
    }

    @Override
    public int getItemCount() {
        return listPDFs.size();
    }
}
```

CÓDIGO FUENTE:

ViewHolderCargamento

```
package com.example.borradoruno.ui.adapter;

import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.borradoruno.datos.Cargamento;
import com.example.borradoruno.R;

/**  Clase que controla una fila del listado */
public class ViewHolderCargamento extends RecyclerView.ViewHolder {

    // Controles del viewholder
    private TextView textViewItemCargamento_Nombre;
    private TextView  textViewItemCargamento_Peso;
    private ImageButton imageButtonItemCargamento_Delete;

 // Interface o normas que debe cumplir la clase que va a recibir eventos desde la
tabla
    public interface CargamentoListener {
        void onCargamentoDelete(Cargamento cargamento);
        void onCargamentoShowInfo(Cargamento cargamento);
    }
    private CargamentoListener cargamentoListener;

  /**  Constructor del viewholder, crea una fila para un cargamento */
    public ViewHolderCargamento(@NonNull View itemView, CargamentoListener
cargamentoListener) {
        super(itemView);

        this.cargamentoListener  = cargamentoListener;

        textViewItemCargamento_Nombre  =
itemView.findViewById(R.id.textViewItemCargamento_Nombre);
        textViewItemCargamento_Peso =
itemView.findViewById(R.id.textViewItemCargamento_Peso);
        imageButtonItemCargamento_Delete =
itemView.findViewById(R.id.imageButtonItemCargamento_Delete);
    }

    /**  Mostrar un cargamento en la fila que indique el adapter */
    public void showCargamento(final Cargamento cargamento) {

  //  Mostrar propiedades del cargamento
textViewItemCargamento_Nombre.setText(cargamento.getDenominacionTactica());
        textViewItemCargamento_Peso.setText(String.format("%.2f",
cargamento.getPeso()) + " Tm");

        //  Eventos
        imageButtonItemCargamento_Delete.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {

                cargamentoListener.onCargamentoDelete(cargamento);
            }
        });
    }
}
```

ViewHolderPDF

```
package com.example.borradoruno.ui.adapter;

import android.content.Context;
import android.view.View;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.borradoruno.R;
import com.example.borradoruno.utils.PDFUtils;
import com.karumi.dexter.Dexter;

public class ViewHolderPDF extends RecyclerView.ViewHolder {

    private TextView textViewItemPDF_Name;

    private Context context;

    public ViewHolderPDF(@NonNull View itemView, Context context) {
        super(itemView);

        this.context  = context;

        textViewItemPDF_Name  =
itemView.findViewById(R.id.textViewItemPDF_Name);
    }

    public void showPdf(final String pdfName) {
        textViewItemPDF_Name.setText(pdfName);

        textViewItemPDF_Name.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {

                new PDFUtils(context, "pdf/" + pdfName,
pdfName).execute();
            }
        });
    }

}
```