Tráfico de contenedores

José Ramón Cabanillas García
Grado de Ingeniería Informática
Desarrollo aplicaciones dispositivos móviles (Android)

Profesor colaborador: David Escuer Latorre
Profesor colaborador: Jordi Almirall López

Profesor responsable: Carles Garrigues Olivella

05/06/2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Tráfico de contenedores*
Nombre del autor:	*José Ramón Cabanillas García*
Nombre del consultor/a:	*David Escuer Latorre* *Jordi Almirall López*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Desarrollo aplicaciones dispositivos móviles (Android)*
Idioma del trabajo:	*Español*
Palabras clave:	*Contenedor, Android*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El tráfico portuario es un sector muy importante para la economía y millones de toneladas de mercancías son movidas en barcos todos los años. Este proyecto es la solución al problema de medir con exactitud el tráfico de contenedores producido en la terminal del Puerto de Melilla, lo que ayudará a la dirección de esta organización a la toma de decisiones. Permite autenticarse al conductor de grúa o de carretilla elevadora que realizará las operaciones. Con cada contenedor que se mueva se insertan la matrícula, el origen, el destino, el consignatario y si está vacío. Todos los pasos se hacen de forma ágil y sencilla, ya que en el desarrollo se ha empleado el diseño centrado en el usuario para dispositivos móviles. El movimiento de un contenedor queda registrado en una base de datos interna. Al finalizar las operaciones, la información se envía desde la app a un servidor virtual central para su análisis, gestión y explotación.

El resultado de esta actividad es una aplicación para dispositivos Android que se instala en una tablet. El conductor es el encargado de introducir toda la información desde su puesto de trabajo y puede corregirla fácilmente. Este proyecto tiene un diseño totalmente centrado en el usuario final, pues se utiliza en un entorno de riesgo, en el que hay que tener gran concentración y velocidad de ejecución. Tras la adquisición directa de datos, se tiene un control exacto de los movimientos producidos en la terminal de contenedores de la Autoridad Portuaria de Melilla.

Abstract (in English, 250 words or less):

Port traffic is a very important sector for the economy and millions of tons of goods are moved by ships every year. This project is the solution to the problem of accurately measuring the container traffic produced at the Port of Melilla terminal, which will help the management of this organization to make decisions. It allows authentication of the crane or forklift driver who will perform the operations. With each container that is moved, the registration number, the origin, the destination, the consignee and if it is empty are inserted. All steps are done in an agile and simple way, since user-centered design for mobile devices has been used in the development. The movement of a container is recorded in an internal database. At the end of the operations, the information is sent from the app to a central virtual server for analysis, management and exploitation.

The result of this activity is an application for Android devices that is installed on a tablet. The crane operator is responsible for entering all the information from his workplace. If an error occurs in any field, it can be corrected easily. This project has a design totally focused on the end user, since it is used in a risk environment, in which it is necessary to have great concentration and speed of execution. After direct data acquisition, there is an exact control of the movements produced in the container terminal of the Port Authority of Melilla.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1. Contexto y justificación del Trabajo

La competitividad reinante en este mundo tan globalizado obliga a las empresas a disponer de toda la información que les sea posible conseguir. Una vez analizada, se obtendrán datos que ayudarán a la toma de decisiones. Esto también ocurre en el sector portuario, donde el proceso de la gestión de mercancías en un puerto es un tema muy importante, que puede ser desconocido para parte de la población, pero que resulta un pilar fundamental tanto para multitud de ciudades que lo trabajan como para la economía en general.

Este Trabajo de Fin de Grado procura resolver un problema que existe actualmente en la Autoridad Portuaria de Melilla (APML), que posee una terminal de contenedores y desea controlar de manera precisa los movimientos de estos producidos en dicha terminal. En la actualidad, esta información se consigue de forma un tanto “manual o artesanal”, basándose en diversos documentos como son las solicitudes de declaración de mercancías de los consignatarios de buques. También se revisan informes tanto por parte de la Guardia Civil como de la Policía Portuaria y Aduanas. Así se obtienen las estadísticas con las que se trabaja.

Tras realizar varias búsquedas a través de la web y en la *Google Play*, no se ha localizado ninguna herramienta o *app* que sea similar y realice la función que se desea conseguir. Puede deberse a la especificidad de la aplicación que ocupa este trabajo, por lo que se trata de un desarrollo totalmente nuevo. Es cierto que hay una herramienta llamada mobilePORT que trabaja con información general de mercancías y atraques de buques, aunque solamente a título informativo (puerto origen, puerto destino, horarios…), aparte de requerir registro para emplear todas sus funciones y de centrarse en el tráfico portuario mexicano. Asimismo, existen multitud de utilidades disponibles en la tienda de Google y relacionadas con el sector marítimo (si bien no con el tema que nos ocupa) como aFerry - Todos los ferrys, CALT App, Posidonia SmartPort, etc.

Se necesita disponer de una gestión más exhaustiva del tráfico de contenedores en el Puerto de Melilla. Para ello, se desarrollará una aplicación que permita llevar un control sobre los movimientos de estos que se realizan en la propia terminal de contenedores. Así pues, tanto la Dirección como el Departamento de Explotación y Planificación de la APML obtendrán la información de primera mano, de manera fiable y precisa, evitando posibles desvíos o erratas que se pudieran producir en los datos recibidos de los distintos actores participantes en el proceso.

1.2. Objetivos del Trabajo

El objetivo general de este TFG es ayudar a la Autoridad Portuaria de Melilla al análisis del tráfico de contenedores producido en su terminal. Los datos obtenidos serán una base fundamental en la que se apoyará la Dirección de la entidad para la toma de decisiones.

Esquemáticamente se resume en los siguientes puntos:

- Proporcionar una herramienta dedicada a registrar el movimiento de contenedores en la terminal de contenedores del Puerto de Melilla.

- Enviar la información registrada en la aplicación a una base de datos remota, ubicada en un servidor virtual de la APML.

- Mejorar la gestión en la explotación del Puerto de Melilla, para lo que se precisa disponer de todos los datos posibles al alcance y que estos sean de calidad. De ahí que se debe identificar y tratar adecuadamente la información, que ha de entenderse como un recurso estratégico.

Una vez registrados y exportados los datos al equipo remoto, serán tratados por un software externo llamado *QlikView*, que permitirá el estudio en profundidad de la información recibida, de forma que será posible mostrar las diferentes gráficas, tablas y cuadros estadísticos necesarios para tal fin. La configuración y uso del programa citado escapa del ámbito de este TFG.

Adicionalmente, existen una serie de objetivos personales:

- Aprender a realizar y gestionar un proyecto a través del Trabajo de Fin de Grado, que permitirá emplear las capacidades adquiridas a lo largo de los años de estudio, además de obtener experiencia y conocimientos que serán aplicables en la vida real.

- Conocer y utilizar el entorno *Android Studio* [1] y sus herramientas para la programación de aplicaciones futuras.

- Desarrollar nuevas habilidades, aptitudes y destrezas, útiles y enriquecedoras para el futuro de cualquier persona.

Desde otro punto de vista, los objetivos se pueden clasificar en funcionales y no funcionales:

1.2.1. Objetivos funcionales

Requisito funcional	Descripción
RF1	Autenticación de usuarios que utilizarán la aplicación
RF2	Identificación de un contenedor mediante la validación de su matrícula
RF3	Selección del origen de un contenedor
RF4	Selección del destino de un contenedor
RF5	Marcador que identifica si un contenedor está vacío o lleva carga
RF6	Registro de los distintos movimientos de contenedores
RF7	Borrado de datos introducidos por si se produce algún error al rellenar los distintos campos
RF8	Volcado de datos a servidor virtual remoto al finalizar la sesión de trabajo. Si hay algún problema de conexión, la información permanece en local hasta que se pueda enviar correctamente

Tabla 1: Requisitos funcionales

1.2.2. Objetivos no funcionales

Requisito no funcional	Descripción
RNF1	Creación de una aplicación instalable en dispositivos con sistema operativo Android
RNF2	Construcción y desarrollo de una base de datos local (en dispositivo móvil) que agiliza la inserción de registros
RNF3	Formar e instruir al personal encargado de transportar los contenedores, esto es, conductores de grúas y de carretillas elevadoras
RNF4	Disponibilidad y flexibilidad de la aplicación al permitir trabajar de forma *offline*
RNF5	Eficiencia y seguridad durante el manejo de recursos, pues el trabajo se desarrolla localmente
RNF6	Usabilidad alta al estar dirigida a personal que maneja maquinaria pesada
RNF7	Rendimiento elevado y diseño centrado en el usuario, pues la forma de operar con la *app* es sencilla y rápida por el personal al que va dirigida, así como sus circunstancias de trabajo

Tabla 2: Requisitos no funcionales

1.3. Enfoque y método seguido

Plantear una estrategia para diseñar y desarrollar un producto requiere seguir una metodología de trabajo. Entre todas las posibles opciones se debe elegir aquel enfoque que mejor se adapte a una situación particular, a las necesidades planteadas o a los requisitos a cubrir. Entonces, las posibles estrategias disponibles para escoger son dos: desarrollar una herramienta nueva o modificar alguna que ya existe, mejorándola, ampliando sus funcionalidades u optimizando sus recursos o rendimientos.

Tras el análisis del mercado, además de revisar en profundidad la plataforma oficial de Android para publicar sus aplicaciones, no he conseguido descubrir ninguna *app* que resuelva la problemática planteada. No digo que no haya, sino que comercialmente, no he sido capaz de encontrar nada parecido o de similares características. Cabe la posibilidad de que exista, pero seguramente serán herramientas *ad hoc*, totalmente personalizadas o para entornos muy reducidos o locales, como el de este caso que nos ocupa.

Esta propuesta me ha complacido enormemente y es una doble motivación personal, pues me permitirá presentarla como Trabajo de Fin de Grado y además proporcionará ayuda a mi empresa a cumplir con el objetivo deseado. Por tanto, para conseguir ambos propósitos se ha decidido comenzar este proyecto desde cero. Con esta elección, también pueden satisfacerse de la mejor manera posible los requisitos necesarios, como por ejemplo el diseño centrado en el usuario, que siempre es importante, pero en esta ocasión cobra una envergadura trascendente, ya que las condiciones en las que se utilizará la herramienta serán especiales y delicadas.

1.4. Planificación del Trabajo

A continuación, se describen los recursos iniciales para realizar el producto final:

- Equipo portátil con sistema operativo *Windows 10* y arquitectura de 64 bits.
- Entorno de desarrollo integrado (IDE) [2] *Android Studio* 3.6.1, que es el oficial para la realización de aplicaciones para Android.
- Procesador de textos *Word* perteneciente al paquete ofimático *Microsoft Office 365*.
- Aplicación *Project 2019* de *Microsoft* para realizar el diagrama de Gantt.
- Tablet Samsung Galaxy Tab A (2016).

- Solución de almacenamiento en la nube *OneDrive* de *Microsoft* para mantener toda la documentación del proyecto sincronizada, en caso de avería o desastre.

Día de la semana	Número de horas mínimo
Lunes	3
Martes	3
Miércoles	3
Jueves	3
Viernes	3
Sábado	3
Domingo	3

Tabla 3: Dedicación estimada al TFG

Si se realiza el cálculo, da un total aproximado de 312 horas. Obviamente se trata de una estimación, porque puede que haya días que se dedique más tiempo y otros en los que no. Los fines de semana y festivos, si es posible, se pueden incrementar aún más las horas de trabajo.

Nombre de actividad	Duración (días)	Fecha de inicio	Fecha de fin
PEC1	**14**	**19/02/2020**	**04/03/2020**
Contexto y justificación. Objetivos. Enfoque y método elegido. Planificación	14	19/02/2020	04/03/2020
Instalación del IDE	1	22/02/2020	23/02/2020
Realización de aplicación inicial	1	03/03/2020	04/03/2020
PEC2	**27**	**05/03/2020**	**01/04/2020**
Diseño y arquitectura	27	05/03/2020	01/04/2020
Diseño centrado en el usuario (DCU)	23	05/03/2020	28/03/2020
Análisis de usuarios y contexto de uso	12	06/03/2020	18/03/2020

Perfiles de usuario	4	09/03/2020	13/03/2020
Diseño conceptual	11	13/03/2020	24/03/2020
Escenarios de uso	2	11/03/2020	13/03/2020
Flujos de interacción	5	14/03/2020	19/03/2020
Prototipado	8	19/03/2020	27/03/2020
Sketches	2	20/03/2020	22/03/2020
Prototipado horizontal de alta fidelidad	3	21/03/2020	24/03/2020
Evaluación	3	25/03/2020	28/03/2020
Diseño técnico	7	25/03/2020	01/04/2020
Diagrama UML de actores y flujo	1	25/03/2020	26/03/2020
Listado de casos de uso	1	25/03/2020	26/03/2020
Diagrama UML de la base de datos y entidad relación	1	26/03/2020	27/03/2020
Diagrama de la arquitectura del sistema	3	27/03/2020	30/03/2020
PEC3	**41**	**02/04/2020**	**13/05/2020**
Desarrollo de la aplicación	41	02/04/2020	13/05/2020
Entrega final	**22**	**14/05/2020**	**05/06/2020**
Finalización del TFG	22	14/05/2020	05/06/2020
Defensa virtual	**4**	**15/06/2020**	**19/06/2020**

Tabla 4: Hitos del TFG

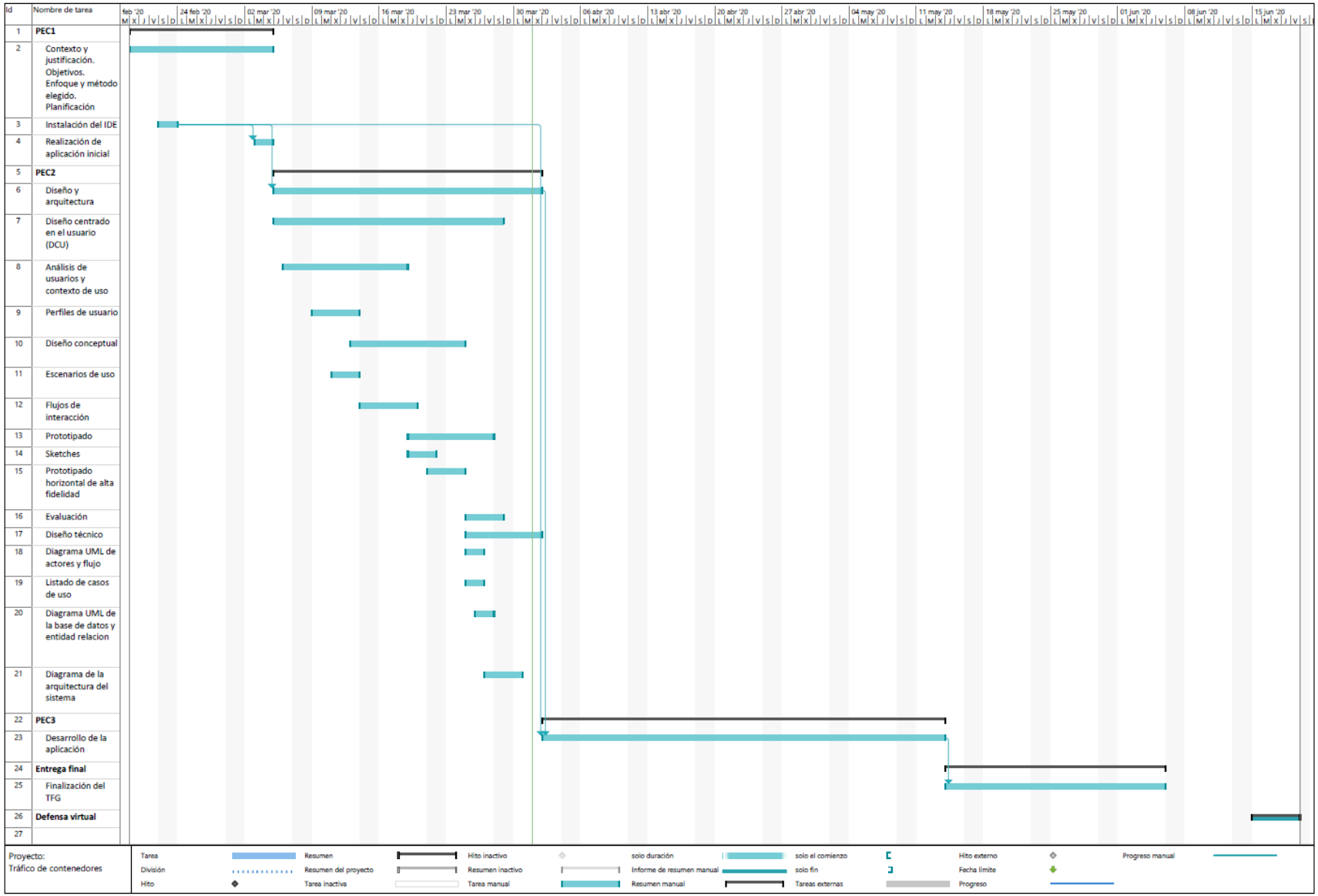


Figura 1: Diagrama de Gantt

1.5. Breve sumario de productos obtenidos

El objetivo final del presente TFG es lograr una *app* para sistema operativo Android que cubra todas las necesidades presentadas por el cliente y que este obtenga la mayor satisfacción posible. A su finalización, se obtendrán los siguientes productos:

- Archivo instalable de la aplicación Android.
- Memoria del Trabajo de Fin de Grado.
- Manual de usuario del producto (anexo a la memoria).
- Código fuente de la herramienta.

1.6. Breve descripción de los otros capítulos de la memoria

Capítulo 2: Diseño y arquitectura
Aplica el Diseño Centrado en el Usuario (DCU) en las fases de análisis, diseño y evaluación de sistemas móviles, que se desarrolla a través de cuatro partes: 1) Usuarios y contexto de uso (análisis). 2) Diseño conceptual (diseño). 3) Prototipado (diseño). 4) Evaluación (evaluación). Aparte de realizar las fases del DCU, también será necesario realizar el diseño técnico de la aplicación. En concreto, se trabajarán dos aspectos del diseño técnico: A. Definición de los casos de uso. B. Diseño de la arquitectura.

Capítulo 3: Desarrollo
Se precisa transformar el diseño realizado en el Capítulo 2 en una aplicación móvil, aprovechando las herramientas apropiadas y así conseguir el objetivo. Para lo cual, se detalla cómo se piensa llevar a cabo, hardware y software empleado para su realización e instalación, decisiones tomadas en su diseño, alternativas existentes y el porqué del camino elegido. Al mismo tiempo, se llevan a cabo pruebas para definir el correcto funcionamiento de la aplicación.

Capítulo 4: Conclusiones
Describen las lecciones aprendidas del trabajo, reflexionan sobre el logro de los objetivos, analizan el seguimiento realizado de la planificación y metodología empleadas y, por último, explora líneas de trabajo futuro que no se han podido investigar.

Capítulo 5: Glosario
Definición de los términos y acrónimos más relevantes utilizados dentro de este documento.

Capítulo 6: Bibliografía
Lista numerada de las referencias bibliográficas utilizadas dentro de la memoria.

Capítulo 7: Anexos
Se proporciona instrucciones de instalación y manual de usuario del producto final realizado.

2. Diseño y arquitectura

2.1. Usuarios y contexto de uso

Los métodos de indagación tienen la finalidad de obtener información en lo referente a las necesidades, objetivos, actitudes y contexto de los usuarios. En el Trabajo de Fin de Grado que nos ocupa, se dan unas circunstancias especiales que nos permiten decantarnos por una técnica de observación e investigación contextual, que tiene la utilidad de conocer el entorno en el que se desarrollan los usuarios y en qué condiciones lo hacen.

Se ha optado por esta técnica porque la aplicación que motiva este proyecto, además de ser *ad hoc*, la utilizará un número extremadamente reducido de sujetos: los cuatro conductores para grúas y carretillas elevadoras que forman parte del personal fijo de la Autoridad Portuaria de Melilla. Debido a que la labor que realiza siempre es la misma, no es relevante el individuo que se escoja para estudiar su entorno, comportamiento y cómo se desenvuelve, pues el resultado de uno de ellos es extrapolable al resto de compañeros. No obstante, tuve la suerte de contemplar a dos conductores en acción.

Para aplicar este método, a pesar de que trabajo en la misma organización, hubo que solicitar permiso al jefe del Departamento de Explotación y Planificación, pues la terminal de contenedores del Puerto de Melilla es una zona de seguridad y restringida que requiere acreditación para el acceso, además de vestimenta de alta visibilidad adecuada, ya que se trata de un lugar peligroso con movimiento constante de grúas, camiones, bateas, personal para la estiba y desestiba, etc. Por consiguiente, me desplacé al lugar donde el usuario utilizará la aplicación. En alguna ocasión anterior y no relacionada con este estudio, ya había tenido la oportunidad de ser testigo de la forma de trabajo en este escenario.

Para el caso genérico de los teléfonos móviles, no suele existir un único entorno en el que los usuarios hagan uso de sus aplicaciones. Sin embargo, en esta ocasión, el marco en el que se desarrolla la acción sí que es siempre el mismo o muy parecido, pues la tarea se lleva a cabo en la misma zona acotada, dentro de la cabina de mando de una grúa o de una carretilla elevadora ocupada por un conductor designado a tal efecto. Lo único que diferencia una intervención de otra puede ser el horario (mañana, tarde o noche) o las condiciones climatológicas.

Tal y como he comentado anteriormente, tuve la suerte de contemplar a dos conductores en acción. Si hubiera podido seguir su evolución junto a alguno de ellos en la cabina, habría sido una experiencia muchísimo más real. Sin embargo, esto no es posible por circunstancias de peligrosidad y seguridad, además de la falta de acondicionamiento de los puestos de conducción para un segundo ocupante. Lo que sí puedo

confirmar es que su ritmo de trabajo es elevado. Afortunadamente, sí que se dio la oportunidad de intercambiar un pequeño diálogo con estas dos personas. Adicionalmente, pude realizar a los cuatro una pequeña encuesta, que me sirvió para crear un perfil de usuario y que se detalla en el siguiente punto.

Como conclusión se obtiene que, la principal petición y/o preocupación del cliente es que la app le reste el menor tiempo posible, que sea ágil y rápida, además de clara, concisa y sencilla. Estas condiciones tienen que cumplirse, pues el consignatario de las mercancías debe abonar la tarifa de grúa según el tiempo que se le dedique a moverlas, es decir, más horas de tráfico de contenedores es mayor cantidad que pagar. Por tanto, el conductor tiene que realizar su trabajo con precaución y celeridad a la vez.

2.1.1. Perfiles de usuario

En este TFG se ha tratado con personas reales, auténticos trabajadores de la Autoridad Portuaria de Melilla, y por la ley de protección de datos, proporcionaré el nombre y las iniciales de los apellidos de los conductores en plantilla que utilizarán esta herramienta.

Usuario	Nombre
A	Antonio C. N.
B	Eduardo C. N.
C	Juan Antonio F. S.
D	Juan Antonio H. C.

Tabla 5: Personal que empleará la utilidad

Respuestas de los usuarios:

Preguntas	Respuestas usuario A
1. Edad	52
2. Ciudad de residencia	Melilla
3. Profesión	Jefe de equipo de mantenimiento
4. Estado civil y número de hijos	Casado. Dos
5. Aficiones	La pesca, la música y mi familia
6. Le interesa Internet y las nuevas tecnologías	Un poco, pero tampoco volverse loco
7. Tiene tablet	Sí, tengo dos

8. Si la respuesta 7. es afirmativa, ¿suele utilizar la tablet?	En casa pero no mucho tiempo, cuando me dejan los hijos. Uso también el móvil
9. Si la respuesta 7. es afirmativa, ¿en qué situaciones prefiere emplear la tablet antes que el móvil?	Para navegar o buscar cosas en Internet y para Netflix

Tabla 6: Respuestas a la encuesta del usuario A

Preguntas	Respuestas usuario B
1. Edad	50
2. Ciudad de residencia	Melilla
3. Profesión	Oficial de mantenimiento
4. Estado civil y número de hijos	Casado. Uno
5. Aficiones	El deporte, correr, nadar, la lectura y escuchar música cuando me dejan
6. Le interesa Internet y las nuevas tecnologías	Sí, me gusta navegar en mi portátil
7. Tiene tablet	Sí
8. Si la respuesta 7. es afirmativa, ¿suele utilizar la tablet?	Cuando estoy en el sofá y no me interesa la TV. También utilizo el móvil para WhatsApp y redes sociales
9. Si la respuesta 7. es afirmativa, ¿en qué situaciones prefiere emplear la tablet antes que el móvil?	Para leer, jugar, ver vídeos, páginas web o compras *online*

Tabla 7: Respuestas a la encuesta del usuario B

Preguntas	Respuestas usuario C
1. Edad	51
2. Ciudad de residencia	Melilla
3. Profesión	Jefe de equipo de mantenimiento
4. Estado civil y número de hijos	Casado. Dos

5. Aficiones	Cocinar para mi familia (cuando puedo), viajar e intentar disfrutar de la vida
6. Le interesa Internet y las nuevas tecnologías	Sí, cuando tengo tiempo me gusta mirar y buscar novedades tecnológicas
7. Tiene tablet	Sí, tengo dos
8. Si la respuesta 7. es afirmativa, ¿suele utilizar la tablet?	A veces, es más cómoda de ver que el móvil
9. Si la respuesta 7. es afirmativa, ¿en qué situaciones prefiere emplear la tablet antes que el móvil?	Para jugar con los niños, navegar, ver televisión, YouTube o leer prensa

Tabla 8: Respuestas a la encuesta del usuario C

Preguntas	Respuestas usuario D
1. Edad	50
2. Ciudad de residencia	Melilla
3. Profesión	Encargado de mantenimiento
4. Estado civil y número de hijos	Casado. Dos
5. Aficiones	Las motos, pescar y viajar
6. Le interesa Internet y las nuevas tecnologías	Sí, utilizo el ordenador de casa para navegar por Internet y buscar gadgets
7. Tiene tablet	Sí
8. Si la respuesta 7. es afirmativa, ¿suele utilizar la tablet?	En ocasiones leo cuando me dejan los hijos. Uso el móvil para redes sociales
9. Si la respuesta 7. es afirmativa, ¿en qué situaciones prefiere emplear la tablet antes que el móvil?	Para navegar y comprar en Amazon o AliExpress, porque se ve mejor que en la pantalla del móvil. También para ver series

Tabla 9: Respuestas a la encuesta del usuario D

Características del perfil

Los perfiles se utilizan para conocer a los usuarios, ya que son agrupaciones de estos según rasgos comunes. Tras el análisis de los datos recopilados, además de tener en cuenta las circunstancias tan especiales que se dan, puesto que inicialmente esta aplicación se va a llevar a cabo para tan solo cuatro personas, se ha desarrollado un único perfil al que se llamará de mediana edad y cuyas características son:

- Edad de 50 a 52 años.
- Residentes en Melilla
- Son compañeros de trabajo y amigos desde hace varios años.
- Su puesto de trabajo es fijo y estable.
- Son personas centradas y responsables, con familia a su cargo.
- Tienen jornada laboral a tiempo completo. Cuando se requiere de alguno/s para mover contenedores, si se produce fuera de horario, realizan horas extra.
- Todos muestran cierto interés en las TIC y poseen móviles y tabletas. Se desenvuelven relativamente bien en cuanto a su utilización.
- En general, eligen las tabletas en casa como dispositivo preferido y están familiarizados con su uso.
- Se observa que forman un grupo bastante homogéneo.

Contexto de uso

Como escenario, hay que indicar que la aplicación se usará en una cabina de conducción de una grúa o de una carretilla elevadora, cuando necesariamente haya tráfico de contenedores, bien en la estiba o en la desestiba de un buque o cuando entran o salen en bateas de camión, pues es necesario introducir los datos del contenedor que se mueve y hacia dónde, y eso debe realizarlo el propio operario. El entorno es dinámico, en constante movimiento, puesto que al trasladar mercancía, se debe manipular con precaución y rapidez, de forma concisa y segura. Las operaciones se pueden llevar a cabo a cualquier hora del día o de la noche, no existen horarios fijos en este contexto ni días concretos de trabajo. Es un servicio disponible todo el año, en consecuencia, centrándose en las condiciones meteorológicas, puede darse calor, frío, viento, lluvia o una mezcla de ellos. En conclusión, no se trata de un contexto de uso tranquilo ni estático, sino todo lo contrario: ajetreado, dinámico, en un horario y clima totalmente aleatorio y con desplazamientos constantes.

Análisis de tareas

Para operar correctamente con la aplicación se deben realizar las siguientes tareas y subtareas:

1. Autenticación en la herramienta.
   1.1. Insertar el nombre de usuario del trabajador.
   1.2. Introducir su contraseña.
2. Gestión de las operaciones.
   2.1. Seleccionar al consignatario de la mercancía.
   2.2. Elegir origen del contenedor (camión, pila o barco).
   2.3. Elegir destino del contenedor (camión, pila, o barco).
   2.4. Introducir su matrícula.
   2.5. Marcar si el contenedor está vacío o no.
   2.6. Guardar el registro.
3. Eliminar el registro de una operación.
   3.1. Sobre el registro deseado, pulsar botón de eliminar.
   3.2. Pulsar botón para confirmar la acción.
4. Finalizar sesión.
   4.1. Pulsar botón para dar fin a la sesión de trabajo.

Listado de características descubiertas gracias a la fase de indagación

- Interfaz clara y despejada, lo más limpia posible. Esta particularidad se aplicará en todas las pantallas que se diseñen.
- Realizar gestión de operaciones en solo lugar.
- Separar elementos (botones, selectores, textos, etc.) todo lo posible entre ellos para evitar pulsaciones accidentales.

2.2. Diseño conceptual

2.2.1. Escenarios de uso

Por lo especial de este Trabajo de Fin de Grado, cuya herramienta principal está inicialmente enfocada a que se utilice por un número muy limitado de personas, es una oportunidad única porque en vez de tratar con personajes se hará directamente con los clientes finales, es decir, se interactúa con los cuatro protagonistas. Tal y como se ha mencionado en

puntos anteriores, el 100% de los usuarios tienen edades comprendidas entre los 50 y los 52 años, nacidos y residentes en Melilla. Están casados y con familia a su cargo, por lo que dan sensación de personas responsables y equilibradas. Muestran interés en las TIC y todos poseen tanto smartphone como tablet. Se defienden en su manejo y son capaces de realizar múltiples tareas como navegación web, compras en tiendas online, uso de redes sociales, ver la televisión, instalar y utilizar diversas aplicaciones, juegos o leer la prensa.

El escenario de uso es sentado en una cabina de mandos, pilotando en unas ocasiones una carretilla elevadora y en otras una grúa. La situación es de concentración, con movimientos constantes en la reubicación de los distintos contenedores, realizando el desplazamiento con celeridad y precisión. El trabajo siempre se lleva a cabo en la misma localización física, que es la explanada de contenedores del Puerto de Melilla. Se emplea una tablet en el puesto de conducción que va fijada a un soporte, así se evitan caídas, golpes o desplazamientos del aparato. Lo que hace el escenario diferente de una ocasión a otra es, por un lado, el horario en el que se realiza y por otro, las condiciones climatológicas. Estos dos puntos son totalmente aleatorios. Al fijarse en el primero, un barco puede atracar o desatracar a cualquier hora del día o de la noche. En cuanto al segundo, las operaciones se hacen en cualquier estación del año, lo que conlleva calor, frío, viento, lluvia o una mezcla de ellas. Estas son tenidas en cuenta por el conductor, aunque al ir siempre a resguardo en el puesto de mando, no le suele resultar desagradable. La condición meteorológica que más les afecta son las fuertes rachas de viento, que pueden dar lugar a cierto movimiento del barco que se trata de cargar o de descargar. En cualquier caso, todos son trabajadores muy experimentados pues el que menos experiencia tiene, lleva ya más de 15 años realizando esta labor.

2.2.2. Flujos de interacción

En esta ocasión se utilizan los flujos de interacción, una herramienta que muestra el camino a seguir por los distintos menús y opciones de la aplicación objeto de este TFG. Se detallan los pasos iniciales para usarla: primero, la pantalla de presentación; segundo, la autenticación en la app; tercero, llevar a cabo la correcta contabilización de los diferentes movimientos de contenedores en la explanada del Puerto de Melilla; cuarto, eliminar un registro erróneo; por último, cerrar adecuadamente la sesión de trabajo. Además, se aportan distintas variantes y caminos según el usuario vaya interactuando con la herramienta, esto es, se exploran todas las vías ofrecidas.

Leyenda

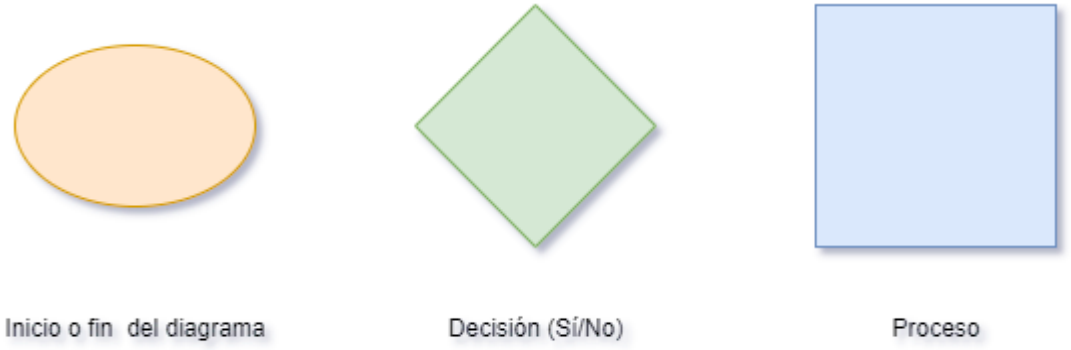


Figura 2: Leyenda del flujo de interacción

Presentación de la aplicación

Figura 3: Presentación de la aplicación

Autenticación de usuario

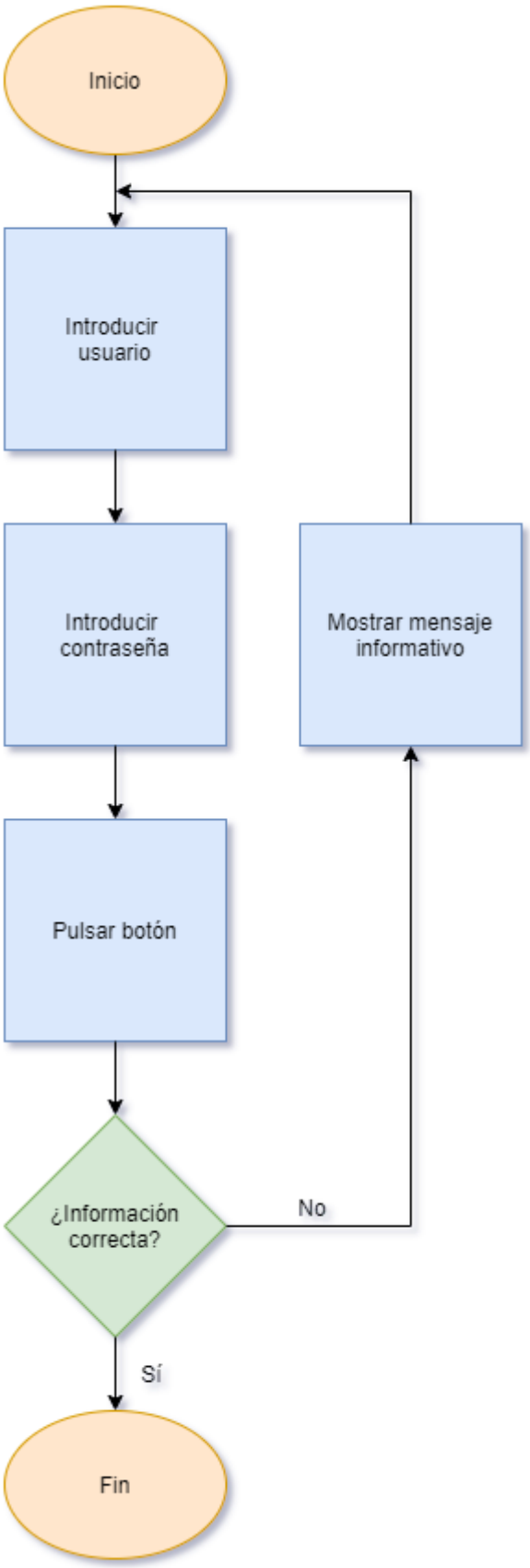


Figura 4: Autentificación de usuario

Registro de operaciones

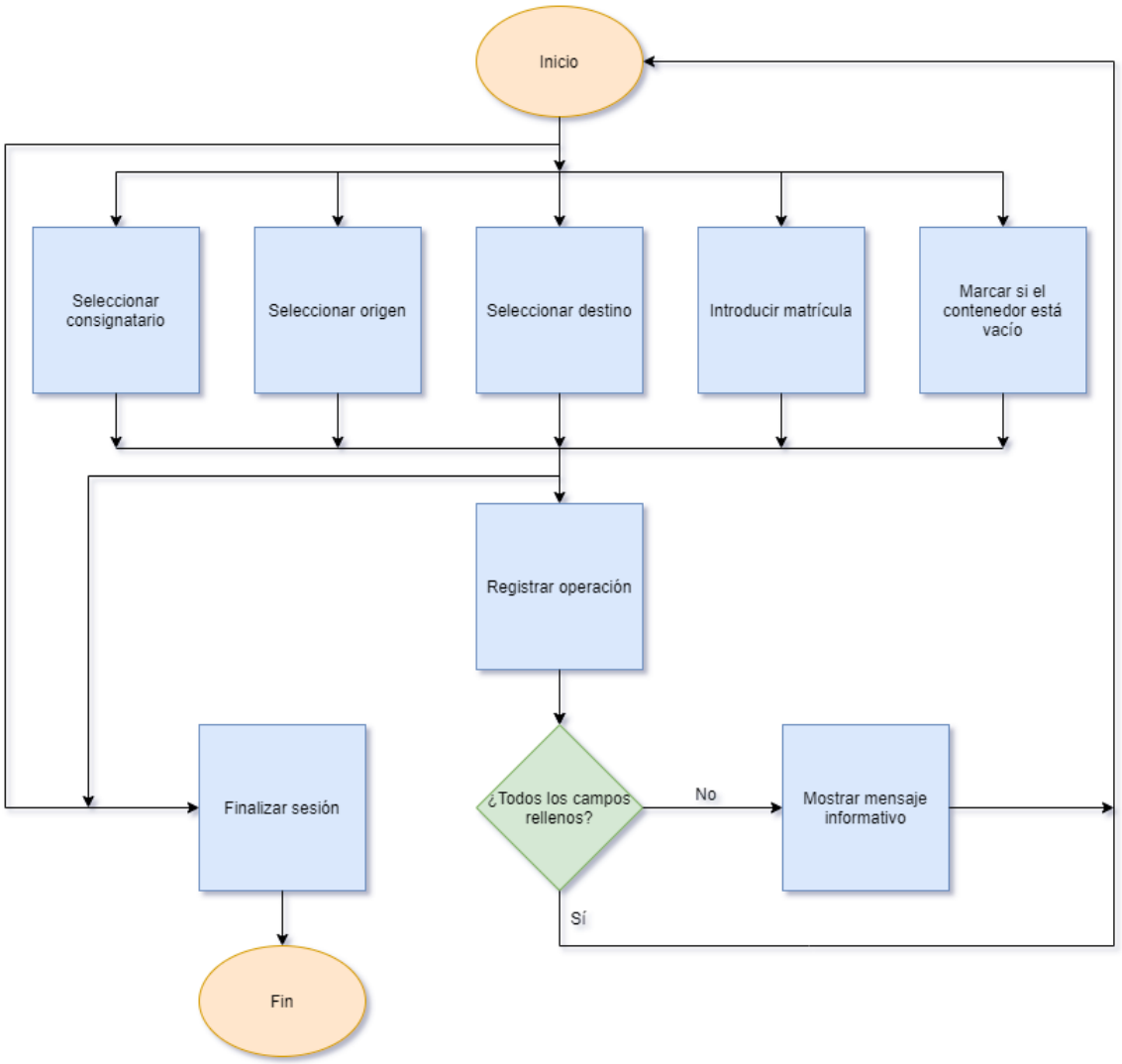


Figura 5: Registro de operaciones

Eliminación de registro

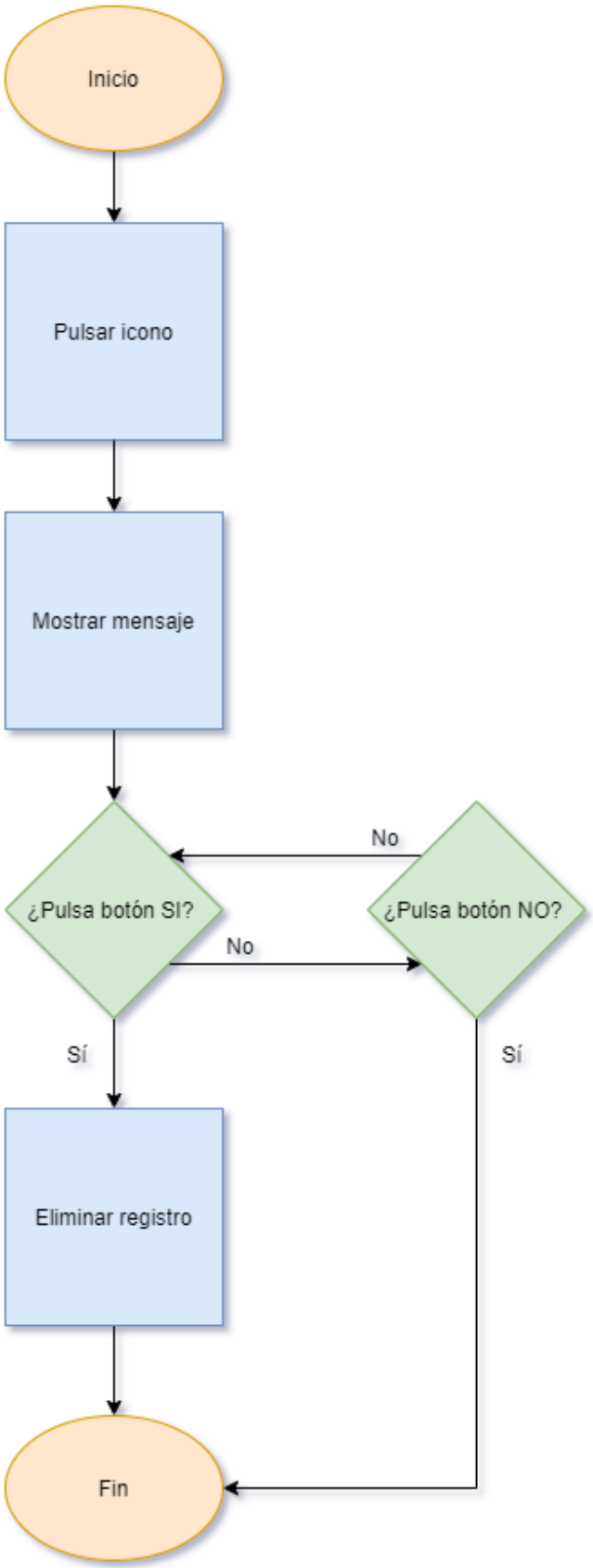


Figura 6: Eliminar registros

2.3. Prototipado

En este momento ya se tienen establecidos los distintos flujos de interacción, con los pasos a seguir en cada uno de ellos. Ahora toca plasmar el trabajo en un prototipo de interfaz, para así obtener un primer acercamiento a la herramienta deseada. Se van a mostrar distintos bocetos genéricos, sin entrar en detalle ni en todas las opciones posibles, que darán una idea de lo que se quiere desarrollar.

Para realizar los prototipos de bajo nivel o “sketches” se utiliza la herramienta en línea *InVision*: https://www.invisionapp.com/ [3]

2.3.1. Sketches

Portada de la aplicación

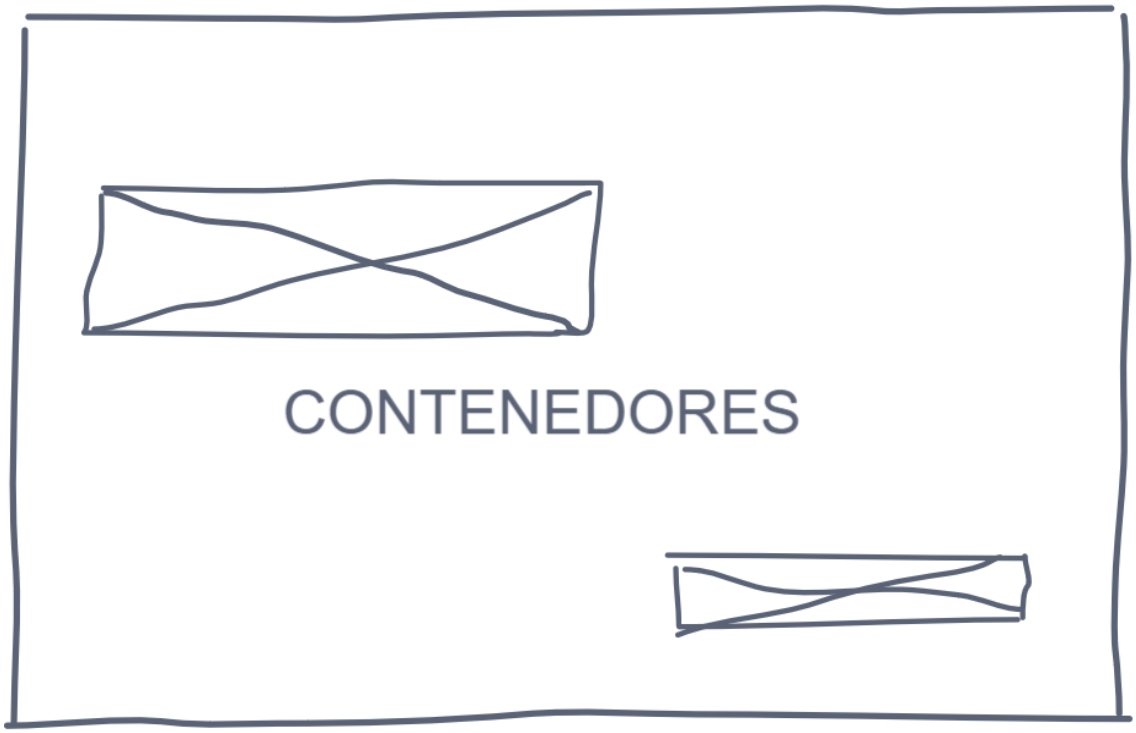


Figura 7: Sketch de la portada

Autenticación de usuario

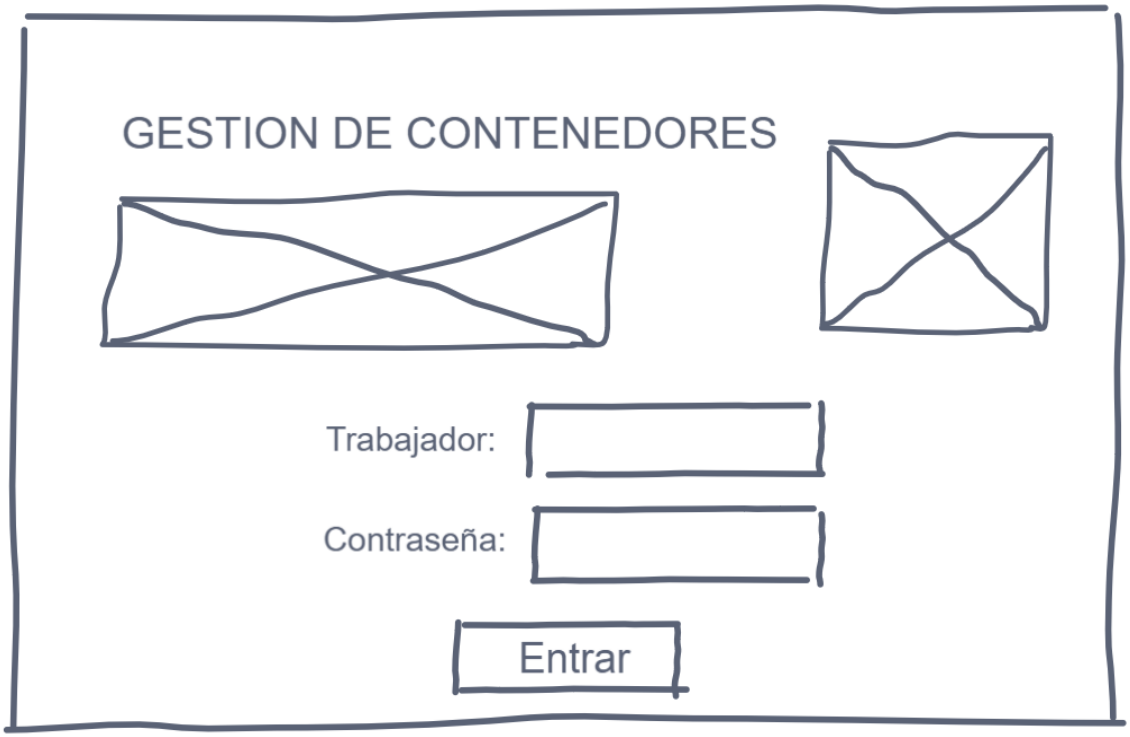
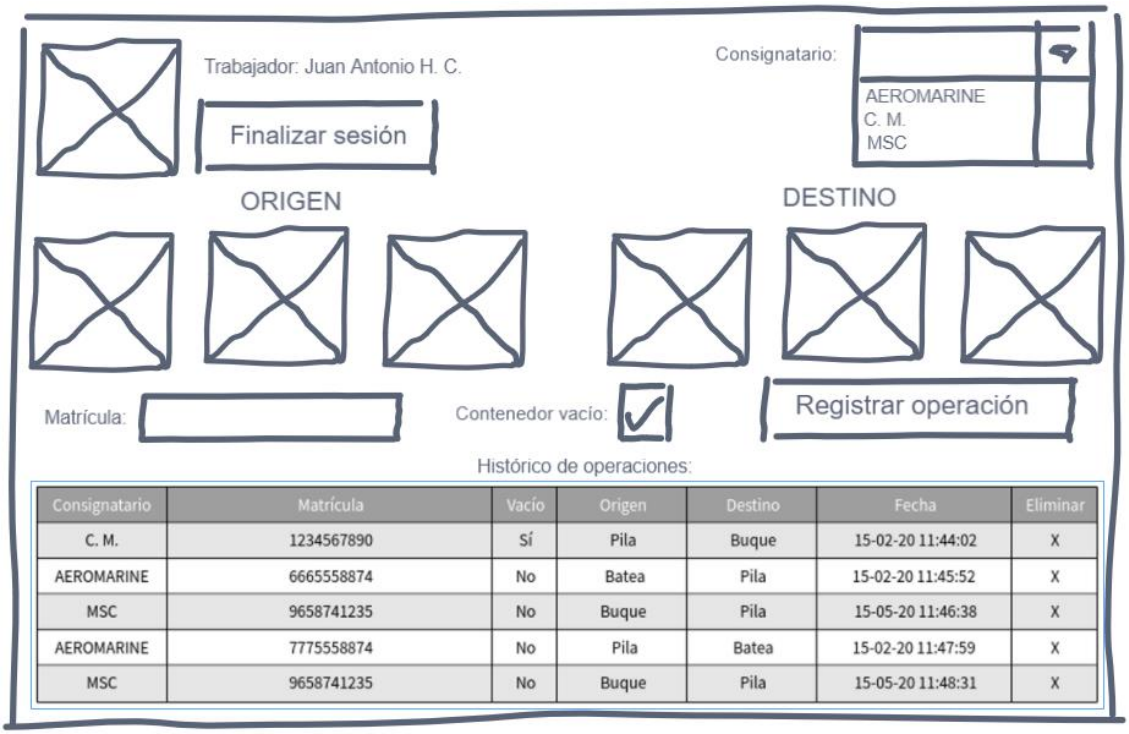


Figura 8: Sketch de autenticación de usuario

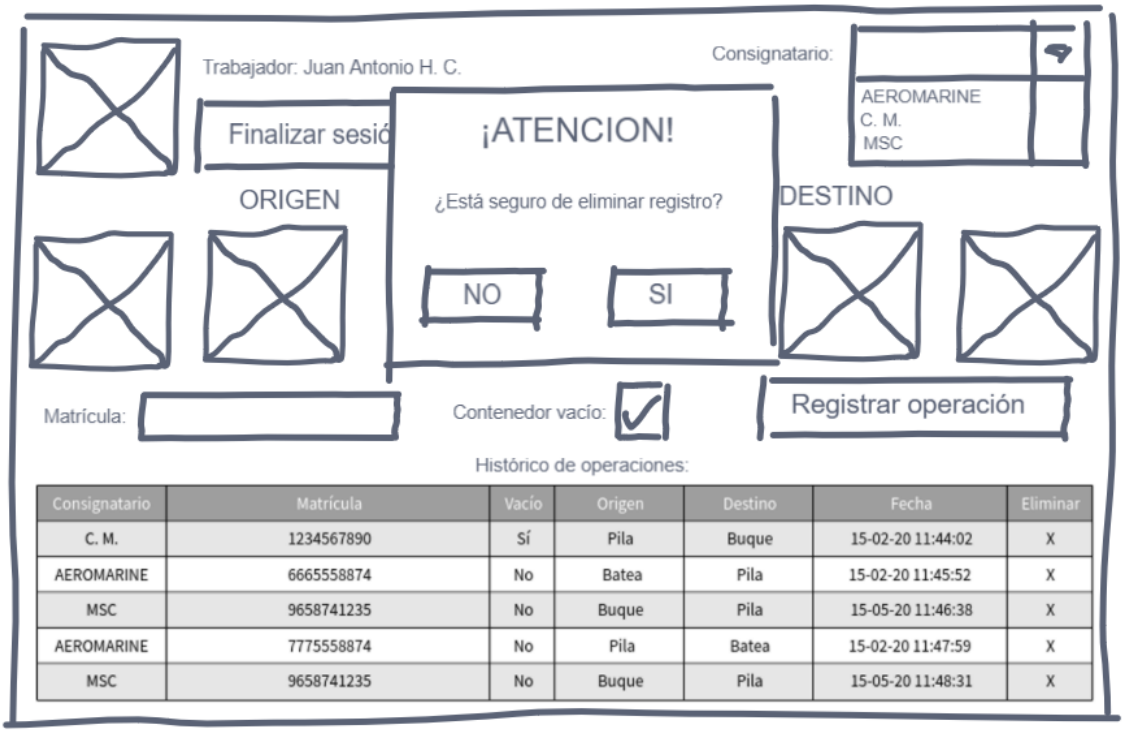
Gestión de las operaciones



Consignatario	Matrícula	Vacío	Origen	Destino	Fecha	Eliminar
C. M.	1234567890	Sí	Pila	Buque	15-02-20 11:44:02	X
AEROMARINE	6665558874	No	Batea	Pila	15-02-20 11:45:52	X
MSC	9658741235	No	Buque	Pila	15-05-20 11:46:38	X
AEROMARINE	7775558874	No	Pila	Batea	15-02-20 11:47:59	X
MSC	9658741235	No	Buque	Pila	15-05-20 11:48:31	X

Figura 9: Sketch de la gestión de las operaciones

Eliminar registro



Consignatario	Matrícula	Vacío	Origen	Destino	Fecha	Eliminar
C. M.	1234567890	Sí	Pila	Buque	15-02-20 11:44:02	X
AEROMARINE	6665558874	No	Batea	Pila	15-02-20 11:45:52	X
MSC	9658741235	No	Buque	Pila	15-05-20 11:46:38	X
AEROMARINE	7775558874	No	Pila	Batea	15-02-20 11:47:59	X
MSC	9658741235	No	Buque	Pila	15-05-20 11:48:31	X

Figura 10: Sketch de la eliminación de registros

2.3.2. Prototipo horizontal de alta fidelidad

Tras desarrollar sketches o interfaces “a mano”, ahora toca realizar prototipos de alta fidelidad, que consisten en la construcción de un modelo de nuestra aplicación que se aproxima todo lo posible a la realidad y permite acercarse con mayor precisión al aspecto final deseado. Gracias a este desarrollo en profundidad, se han corregido errores que pasaron desapercibidos en los dibujos del punto anterior.

Para realizar los prototipos horizontales de alta fidelidad (o de alto nivel) se utiliza el programa *Justinmind Prototyper FREE Edition*, en su versión 8.7.7: https://www.justinmind.com/ [4]

Portada de la aplicación

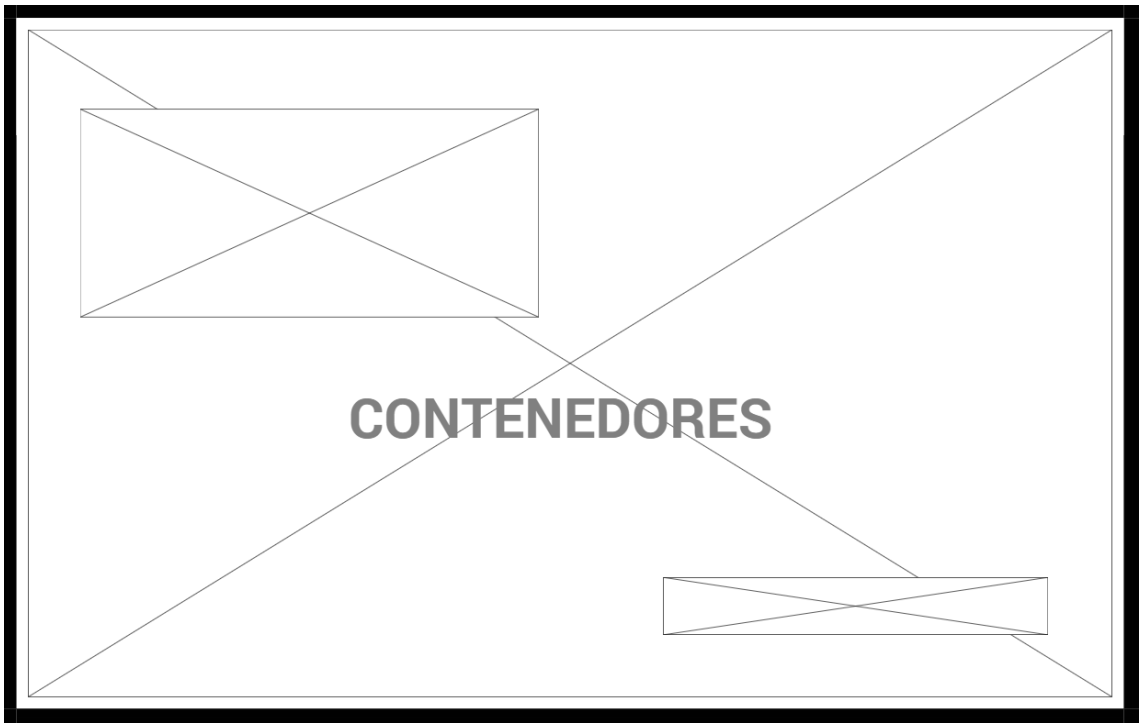


Figura 11: Prototipo de la portada

Autenticación de usuario

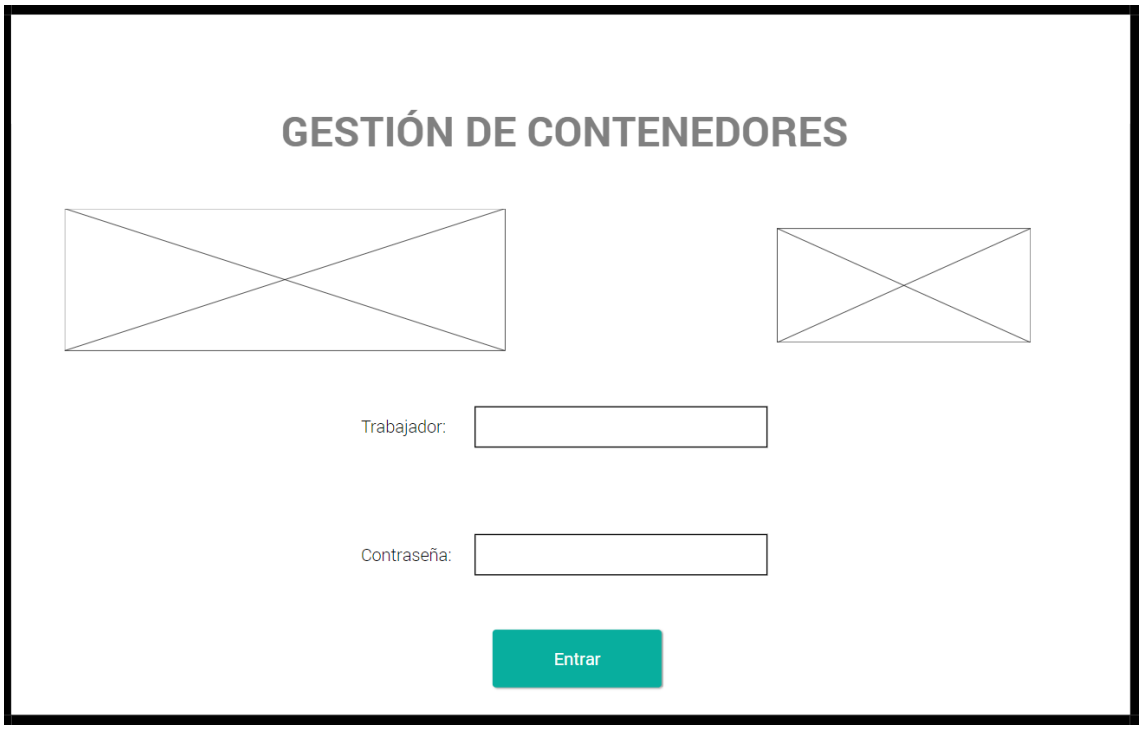


Figura 12: Prototipo de la autenticación en el sistema

Gestión de las operaciones

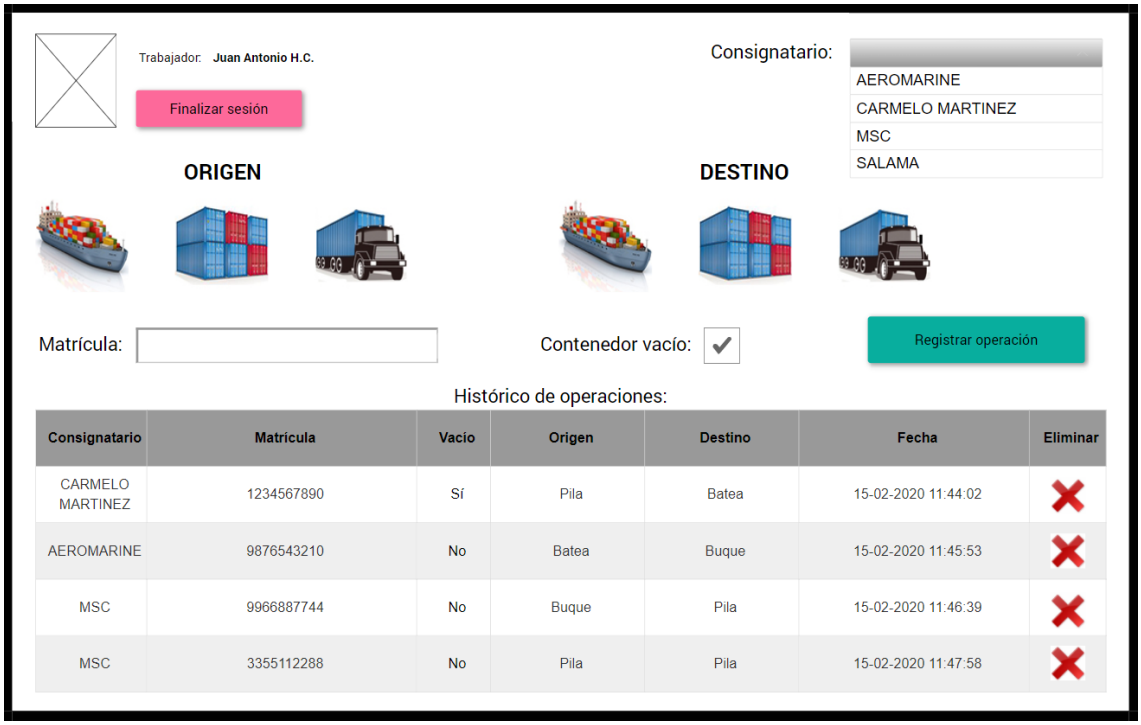


Figura 13: Prototipo de la gestión de operaciones

Eliminar registro

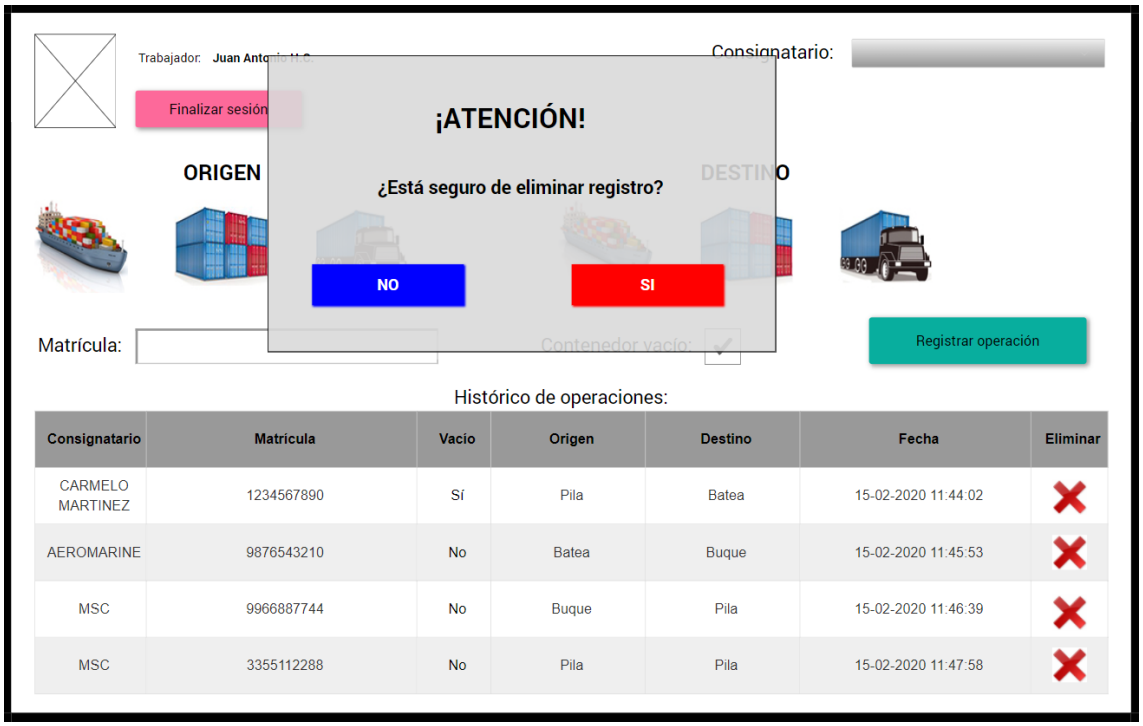


Figura 14: Prototipo para eliminar registros

Explicaciones de las soluciones de diseño propuestas

La portada es una sencilla presentación de la aplicación con una imagen de fondo y sobre ella, un logo y otra imagen además de un texto con el nombre de la herramienta.

La parte de autenticación es una interfaz simple en la que tan sólo hay que introducir nombre de usuario y contraseña para poder emplear la utilidad de tráfico de contenedores.

La pantalla de gestión de operaciones es el corazón de la herramienta, en donde se administrarán las diferentes acciones. Se realiza todo en el mismo lugar, pues de esta manera, el usuario no requerirá avanzar, buscar o retroceder por opciones o menús, puesto que podría llevarle a errores o distracciones totalmente innecesarias, y que consigue que se trate toda la información desde un único sitio. De producirse errores al introducir los datos, es posible eliminarlos posteriormente. Para ello, se pulsa sobre el icono “X” junto al registro que se desea suprimir y se pide confirmación de la operación.

Se ha diseñado de la forma más austera y clara posible, para evitar confusiones o despistes, además de intentar aplicar la máxima simplicidad y agilidad para todas las pantallas. Al mismo tiempo, se tiene siempre presente al usuario como el centro del diseño elegido. De manera adicional, se han intentado seguir las buenas prácticas para el diseño en dispositivos móviles, así como la evaluación heurística, la extrapolación de las ocho reglas de oro de Shneiderman en aplicaciones móviles, la guía heurística para el diseño móvil de Savio y Braiterman, y por último, la guía heurística para la computación móvil de Bertini, Gabrielli y Kimani. En definitiva, se ha implementado la aplicación intentando seguir todas las pautas recomendadas, además de escuchar, respetar y utilizar las opiniones obtenidas de los interesados, que a la postre, son el principal objetivo al que hay que atender y satisfacer.

2.4. Evaluación

El objetivo de la última fase es la planificar la evaluación del prototipo. El DCU es un proceso iterativo y, por tanto, hay que evaluar los diseños y corregir los errores de manera iterativa.

No obstante, para no incrementar la carga de trabajo, en esta fase se indica cómo se llevaría a cabo la evaluación (sin necesariamente realizarla).

2.4.1. Preguntas para obtener información del usuario

	Preguntas
1.	Facilidad de acceso a la aplicación
2.	Valoración sobre la autenticación
3.	Opinión que le merece la interfaz de operaciones
4.	Le gusta
5.	¿Por qué?
6.	Qué mejoraría o cambiaría
7.	Qué suprimiría
8.	¿Le parece una herramienta ágil, clara y veloz?
9.	Impresión general y grado de satisfacción
10.	Si lo desea, exponga cualquier otra apreciación, idea o comentario que se le ocurra

Tabla 10: Preguntas sobre información referente al usuario

2.4.2. Tareas a realizar por los usuarios

	Tareas
1.	Acceder a la aplicación
2.	Autenticarse
3.	Insertar varios registros
4.	Eliminar algún registro
5.	Finalizar sesión

Tabla 11: Tareas que los usuarios deberían realizar

2.4.3. Preguntas referentes a las tareas

	Preguntas
1.	¿Le ha parecido sencillo autenticarse?
2.	¿Qué mejoraría en el apartado de autenticación?
3.	¿Qué le parece la pantalla de gestión de las operaciones?
4.	¿Le resulta cómoda?

5.	**¿Le parece simple y clara?**
6.	**¿Cambiaría algo de tamaño o de posición?**
7.	**¿Eliminaría o añadiría algún objeto? ¿Cuál?**
8.	**¿Es rápida y sencilla la inserción de datos de un contenedor?**
9.	**¿Es útil y de dimensión adecuada la opción de eliminar registro?**
10.	**¿Le parece una herramienta ágil, sencilla y veloz?**
11.	**¿Le satisface en general esta interfaz para las operaciones?**
12.	**¿Desea añadir o aportar alguna otra cosa?**

Tabla 12: Preguntas referentes a las tareas

2.5. Definición de los casos de uso

2.5.1. Diagrama UML de actores y flujo

En este caso el actor, que se corresponde con el usuario de la *app*, se relaciona con el sistema pero es externo a este. La forma de interactuar es a través de los casos de uso expuestos en el diagrama.

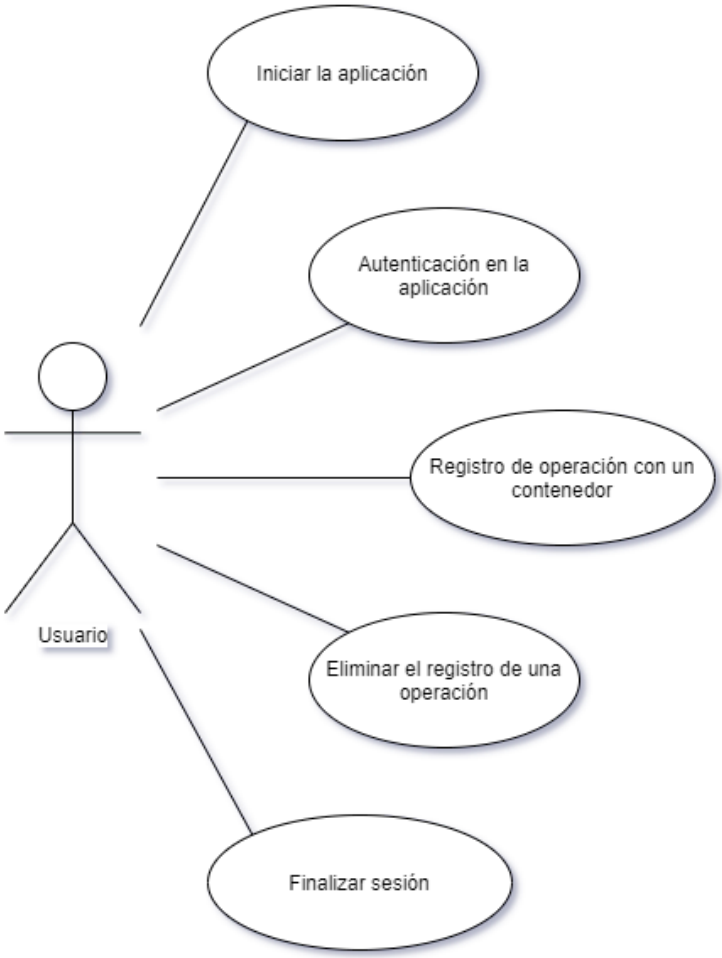


Figura 15: UML de actores y flujo

2.5.2. Listado de los casos de uso

Identificador	**CU-001**
Nombre	Iniciar la aplicación
Prioridad	Normal
Descripción	El usuario ejecuta la aplicación para controlar el tráfico de contenedores
Actores	Usuario
Precondiciones	Se desea iniciar el registro de movimientos de contenedores
Iniciado por	Usuario que gestionará los movimientos de los diversos contenedores
Flujo	El usuario procede a abrir la aplicación
Postcondiciones	Acceso a la sección de autenticación
Notas	Ninguna

Tabla 13: Caso de uso 001

Identificador	**CU-002**
Nombre	Autenticación en la aplicación
Prioridad	Normal
Descripción	Se realiza la autenticación del usuario en el sistema
Actores	Usuario
Precondiciones	Haber iniciado la aplicación (caso de uso CU-001)
Iniciado por	Usuario que quiere autenticarse
Flujo	El usuario introduce su identificador El usuario introduce su contraseña El usuario pulsa botón para entrar
Postcondiciones	Acceso a la sección de gestión de operaciones
Notas	Sin credenciales correctas, no se accede a la siguiente sección y el sistema queda a la espera

Tabla 14: Caso de uso 002

Identificador	**CU-003**
Nombre	Registro de operación con un contenedor
Prioridad	Alta
Descripción	Se realiza el registro del movimiento de un contenedor

Actores	Usuario
Precondiciones	Haber introducido credenciales correctas (caso de uso CU-002)
Iniciado por	Usuario
Flujo	El usuario selecciona consignatario El usuario selecciona origen El usuario selecciona destino El usuario introduce matrícula El usuario marca si el contenedor está vacío El usuario pulsa botón para registrar
Postcondiciones	Registro del movimiento de un contenedor. El sistema queda libre y a la expectativa de la siguiente operación
Notas	Si los datos no son completos, queda a la espera de que se introduzcan

Tabla 15: Caso de uso 003

Identificador	**CU-004**
Nombre	Seleccionar el registro de una operación
Prioridad	Alta
Descripción	Se marca el registro deseado del movimiento de un contenedor para eliminar
Actores	Usuario
Precondiciones	Haber realizado el registro de una operación (caso de uso CU-003)
Iniciado por	Usuario
Flujo	El usuario, en el registro deseado, pulsa botón de eliminar ubicado a la derecha de la tabla (icono rojo en forma de “X”)
Postcondiciones	El sistema queda a la espera de confirmación la acción
Notas	Ninguna

Tabla 16: Caso de uso 004

Identificador	**CU-005**
Nombre	Eliminar el registro de una operación
Prioridad	Alta
Descripción	Se elimina el registro del movimiento de un contenedor
Actores	Usuario
Precondiciones	Haber seleccionado un registro de operación para eliminar (caso de uso CU-004)
Iniciado por	Usuario

Flujo	El usuario pulsa botón “SI” de confirmar eliminación
Postcondiciones	El sistema queda a la espera de más acciones
Notas	Si el usuario pulsa “NO” a la eliminación, el sistema queda libre y a la expectativa de la siguiente operación

Tabla 17: Caso de uso 005

Identificador	**CU-006**
Nombre	Finalizar sesión
Prioridad	Normal
Descripción	Finaliza la sesión del usuario
Actores	Usuario
Precondiciones	Haber realizado el registro de una operación (caso de uso CU-003)
Iniciado por	Usuario
Flujo	El usuario pulsa el botón de finalizar sesión El sistema envía la base de datos local al servidor virtual remoto
Postcondiciones	Se cierra la aplicación
Notas	Si no hay conexión o cobertura, la información queda almacenada localmente hasta el próximo intento

Tabla 18: Caso de uso 006

2.6. Diseño de la arquitectura

2.6.1. Diagrama UML del diseño de la base de datos

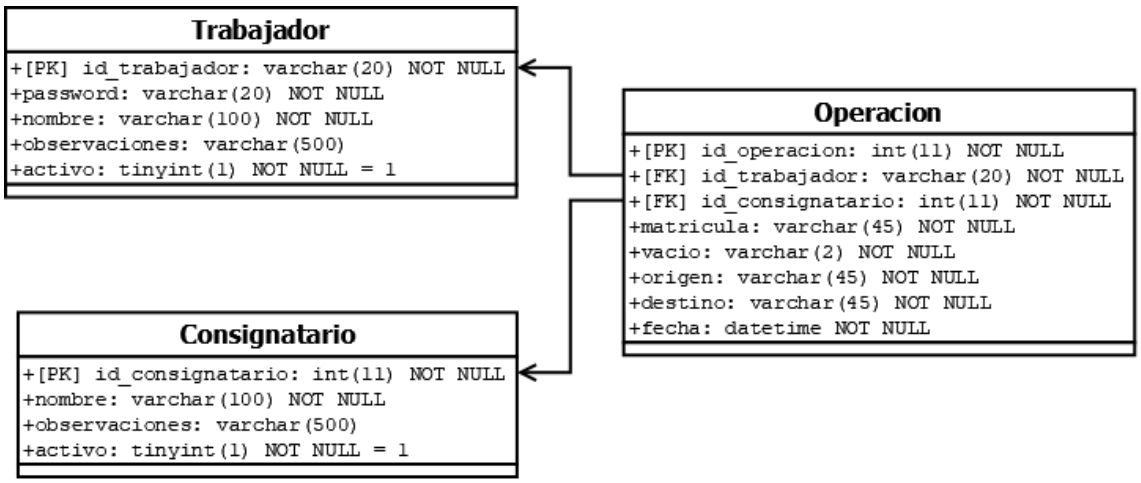


Figura 16: UML de la base de datos

Tabla Trabajador

En esta tabla se registran los distintos operarios que emplearán la utilidad y que podrán autenticarse en ella, así como diversas características de estos.

Campo	Descripción
id_trabajador	Identificador del trabajador. Clave primaria
password	Contraseña del trabajador. No nulo
nombre	Nombre del trabajador. No nulo
observaciones	Por si se requiere alguna observación
activo	Si el trabajador está activo o no. Por defecto se crea activo (1). No nulo

Tabla 19: Tabla Trabajador

Tabla Consignatario:

En esta tabla se registran los diferentes consignatarios de mercancías que operan actualmente en las instalaciones de la Autoridad Portuaria de Melilla. También almacena algunas características de dichos consignatarios.

Campo	Descripción
id_consignatario	Identificador del consignatario. Clave primaria
nombre	Nombre del consignatario. No nulo
observaciones	Por si se requiere alguna observación
activo	Si el consignatario está activo o no. Por defecto se crea activo (1). No nulo

Tabla 20: Tabla Consignatario

Tabla Operación:

En esta tabla se registran todos los movimientos de contenedores producidos y sus características asociadas. Un trabajador puede realizar una, ninguna o múltiples operaciones. Un cosignatario puede consignar un contenedor, ninguno o múltiples. En un momento determinado, un contenedor es trasladado por un único conductor y también consignado por un único consignatario.

Campo	Descripción
Id_operacion	Identificador de operación. Clave primaria
id_trabajador	Identificador del trabajador. Clave foránea. No nulo
id_consignatario	Identificador del consignatario. Clave foránea. No nulo
matricula	Matrícula del contenedor. No nulo
vacio	Si el contenedor contiene carga o no (Si/No). No nulo
origen	Origen del contenedor (Buque/Pila/Tierra). No nulo
destino	Destino del contenedor (Buque/Pila/Tierra). No nulo
fecha	Fecha y hora de la operación. Se guarda en formato ISO *aaaa-mm-dd hh-mm-ss*. No nulo

Tabla 21: Tabla Operación

2.6.2. Diagrama UML entidad relación

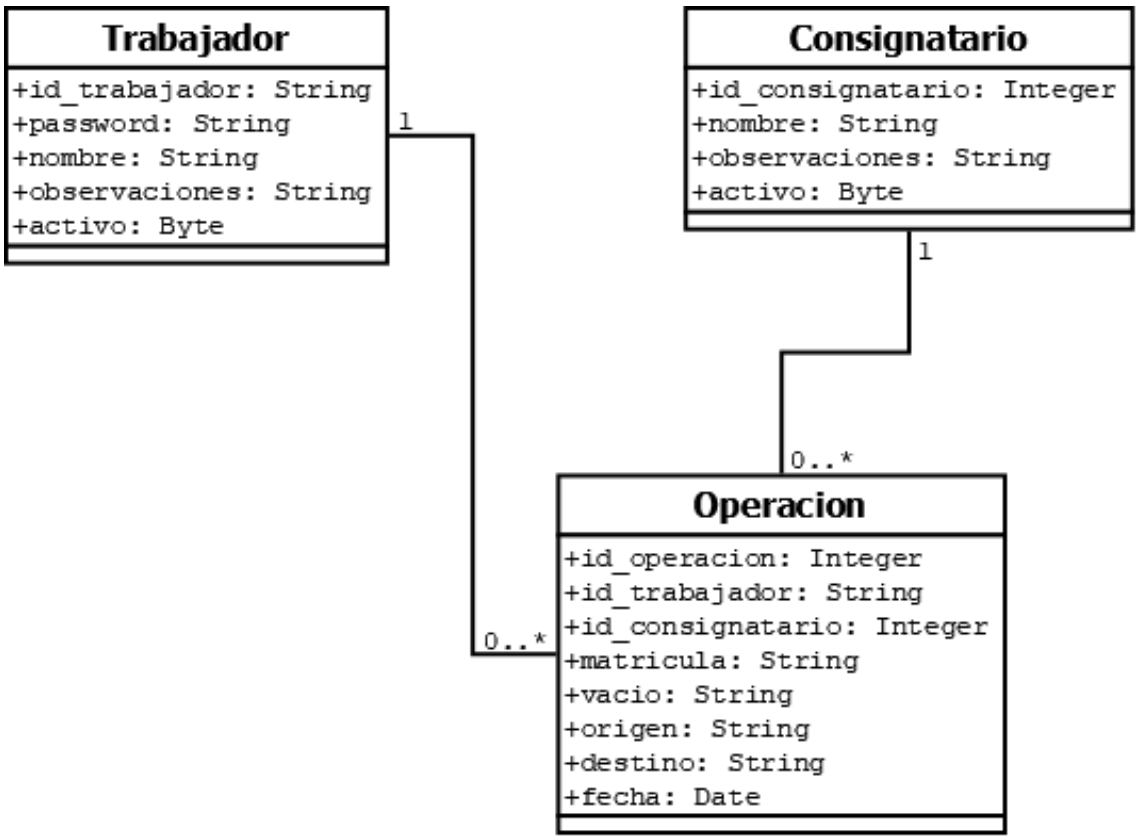


Figura 17: UML del modelo entidad relación

2.6.3. Diagrama de la arquitectura del sistema

La aplicación objetivo de este trabajo se implementa para el sistema operativo Android y será una *app* híbrida; pero ¿qué es eso? Son aplicaciones móviles diseñadas en un lenguaje de programación web (normalmente HTML5, CSS, JavaScript o su combinación), junto con un *framework* que permite adaptar la vista web a cualquier dispositivo móvil. En otras palabras, no es más que una herramienta construida para ser utilizada o implementada en distintos sistemas operativos móviles (como iOS, Android, Windows Phone…), evitando la tarea de crear una *app* para cada uno de ellos. De esta manera, una utilidad híbrida puede ser adaptada a múltiples plataformas móviles sin crear nuevos códigos, tan solo ajustando algunos parámetros y características para cada uno de ellos.

WebView: permite mostrar la interfaz como una página web, es decir, es un navegador integrado en la propia *app*. No incluye las funciones de un *web browser* tradicional y completamente desarrollado, como controles de navegación o una barra de direcciones. De forma predeterminada, todo lo que hace *WebView* sencillamente es mostrar una página web.

Front-end: es la sección que presenta el *WebView* y son páginas web formadas por una combinación de HTML5, CSS3 y JavaScript, que son las tecnologías que integran la parte de interfaz de usuario y nos permiten interactuar con la aplicación y utilizarla. Con ellas se implementa la estructura, el diseño, el comportamiento y el contenido de todo lo que se ve en la pantalla.

Back-end: es el lado donde se almacenan, tratan y organizan los datos. Esta sección no permite que se vea o se interactúe directamente con ella, sino todo lo contrario: los usuarios acceden de forma indirecta a las partes y características del *back-end* a través de una aplicación *front-end*. En este caso, la información se almacena en una base de datos SQLite y se opera con ella a través de Java.

Al finalizar una sesión de trabajo, la información almacenada será enviada a un servidor virtual remoto a través de Internet. Se conservará en una base de datos MySQL para su explotación.

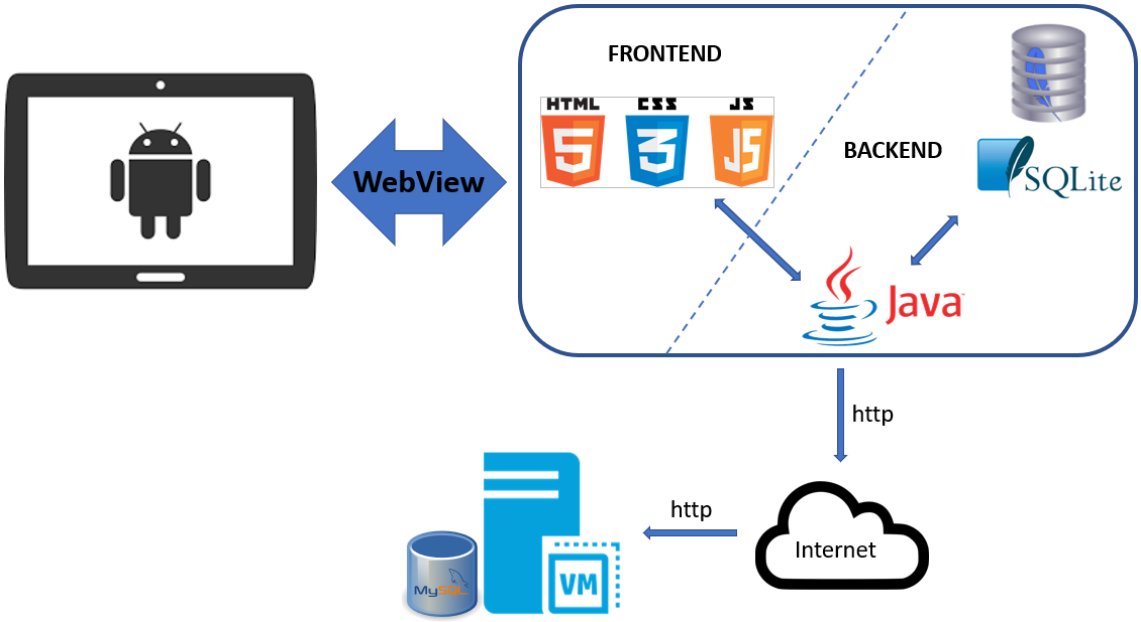


Figura 18: Arquitectura del sistema

3. Desarrollo

3.1. Herramientas utilizadas, lenguajes de programación y APIs

Android Studio [1]

Ha sido la principal herramienta que se ha utilizado para la realización del Trabajo de Fin de Grado, la cual ha evolucionado con diferentes actualizaciones desde su primera instalación (versión 3.5.3), en la última semana de febrero aproximadamente, hasta la que actualmente se emplea en el momento de escribir este punto de la memoria (versión 3.6.3). Es el IDE [2] oficial para el desarrollo de aplicaciones para Android, basado en IntelliJ IDEA. Además del editor de código y las herramientas que contiene para *developers*, ofrece incluso más funciones que aumentan la productividad cuando desarrollas utilidades para Android. Incluye SDK Manager, que es un gestor de APIs y complementos para el entorno. Android Studio permite emplear dos lenguajes de programación, **Kotlin** y **Java**. Se ha decidido emplear la segunda opción porque se conoce mejor y existe más confianza en ella. Kotlin es desconocido para el programador

Además de para el tratamiento y codificación del código Android y sus elementos obligatorios para un funcionamiento adecuado, se ha empleado el IDE para editar y gestionar todo el código fuente de la herramienta, así como los diferentes lenguajes de programación que se han usado en el desarrollo de la aplicación, y que han sido:

- Java
- CSS
- JavaScript
- HTML
- Lenguaje SQL para la creación de sentencias destinadas a la gestión de la base de datos SQLite

Estas razones expuestas son las que consiguieron decantarnos para emplear este IDE. No obstante, aunque Android Studio ha sido el pilar fundamental para la realización del trabajo, ya que nos dispensa todo lo necesario para la correcta confección de la *app*, no es la única herramienta utilizada en este TFG. Como apoyo, se han empleado algunos recursos adicionales que, bien por su facilidad de uso, utilidad, comodidad o por el conocimiento previo del programador sobre ellos, también han resultado importantes para lograr el fin deseado. Seguidamente se enumeran:

Notepad++ (ver. 7.8.6) [5]

Tal y como se ha comentado en el párrafo anterior, la totalidad de la codificación se ha llevado en Android Studio, lo que no quita que, en varias ocasiones, se haya recurrido a este gran editor de licencia GPL, porque ciertas partes del código se examinaban desde otra perspectiva con esta sencilla y potente herramienta. Su interfaz simple y clara, su sistema de búsquedas, el ser multilenguaje así como su sistema de resaltado de palabras o expresiones, ha sido de gran utilidad en las diferentes fases del trabajo.

Https://html-css-js.com/ [6]

Herramienta online con la que se consigue un tratamiento ágil, cómodo y sencillo de los lenguajes HTML, CSS y JavaScript. Ha resultado de enorme utilidad para el desarrollo y depuración de los diferentes documentos HTML que componen la aplicación. Gracias a este sitio se ha conseguido crear, ubicar, aplicar estilos, tamaños, colores, etc. a cada una de las páginas que componen la interfaz de la *app*, así como que sean totalmente interactivos y realicen las acciones deseadas de manera correcta.

DB Browser for SQLite (ver. 3.11.2 portable) [7]

Es un software de código abierto con interfaz cómoda e intuitiva para crear, diseñar y editar archivos de bases de datos compatibles con SQLite. La pueden explotar tanto usuarios como desarrolladores que desean administrar bases de datos. Utiliza una interfaz sencilla pero potente que ha permitido realizar con facilidad y eficacia todas las acciones necesarias a ejecutar en esta aplicación. Ha sido de suma utilidad y se ha empleado para comprobar varias circunstancias relacionadas con la *app* como:

- Creación correcta de las diferentes tablas
- Exploración de su estructura, campos y tipos
- Revisión de las distintas operaciones realizables (creación, selección, inserción y eliminación)
- Comprobación de la apropiada aplicación de instrucciones

3.2. Revisión de la planificación

Finalmente y llegados a este punto, una vez expuesto todo el software empleado para la realización de la aplicación objeto de este TFG, se presentará una contrariedad surgida, que ha producido una pequeña **desviación del proyecto**.

Como bien es conocido por todo el mundo, por desgracia, la pandemia global que padecemos debido al maldito COVID-19 ha afectado de manera drástica al normal desarrollo de nuestras vidas. No es una exageración, es un hecho que todos hemos podido constatar en primera persona. Esta afección ha cambiado drásticamente nuestro quehacer diario, y personalmente, el conjugar la vida privada y familiar, con la laboral y la de estudiante, es una odisea. Por mi profesión, el asunto del teletrabajo ha añadido diversas funciones "extra" a las ya cuantiosas que realizaba, lo que ha generado una subida exponencial de faena. Esta situación -además de las otras dos asignaturas que actualmente curso junto al TFG- me ha llevado a dedicar bastantes más horas de las habituales a mi empleo y familia, por lo que me ha resultado imposible el concluir totalmente la aplicación. Concretamente, ha quedado pendiente la última parte formada por dos apartados relacionados:

1. Finalizar la sesión de trabajo.

2. Enviar los datos recogidos al servidor remoto.

No creo que suponga un problema importante, puesto que para compensar y corregir este hecho, se obtendrá un incremento de tiempo dedicado a este proyecto. La forma de conseguirlo es levantarse una hora más temprano (a veces incluso antes) a diario; esta simple solución lleva ya unos días aplicándose. Asimismo, durante la próxima y última fase "Entrega final", se pulirán todos los fallos y detalles restantes. Concluiremos de forma definitiva con la entrega del código fuente, la memoria y la aplicación finalizadas.

3.3. Implementación

3.3.1. Estructura

En la siguiente tabla se muestran los componentes generales y la estructura de la aplicación Android, así como algunos de los directorios concretos para esta.

Directorio	Fichero	Descripción
/app		Esta es la raíz de la que parte toda la aplicación
/app/libs	mysql-connector-java-5.1.49.jar	Librerías externas necesarias para el funcionamiento de la *app*
/app/release	Contenedores.apk	Se ubica el instalable de la aplicación
/app/src		Contiene todos los códigos

		desarrollados en la utilidad: código fuente y código de pruebas
/app/src/androidTest	MainActivityTest.java	Pruebas instrumentadas (por ejemplo, de interfaz de usuario)
/app/src/test	UserTest.java	Pruebas unitarias
/app/src/main	AndroidManifest.xml	Es el archivo de configuración de toda la aplicación Android. Se define el permiso de acceso a Internet, la actividad inicial, el nombre de la instalación de la *app*, etc.
/app/src/main/assets		Se ubican los diferentes ficheros html que carga la herramienta
/app/src/main/assets/js	funciones.js	Contiene las funciones JavaScript empleadas
/app/src/main/assets/template	estilo.css	Hoja de estilo de las webs de la *app*. Además, contiene las imágenes utilizadas
/app/src/main/java	BaseDeDatosLocal.java JavaScriptInterface.java MainActivity.java	Clases de Java que componen la aplicación
/app/src/main/res		Almacena recursos
/app/src/main/res/layout	activity_main.xml	Aquí se define la interfaz gráfica.
/app/src/main/res/values	strings.xml	Configura el nombre de la aplicación

Tabla 22: Estructura de directorios y componentes de la aplicación

Tras la revisión de la estructura general y los componentes del desarrollo del proyecto, se describen algunos directorios con un poco más de profundidad:

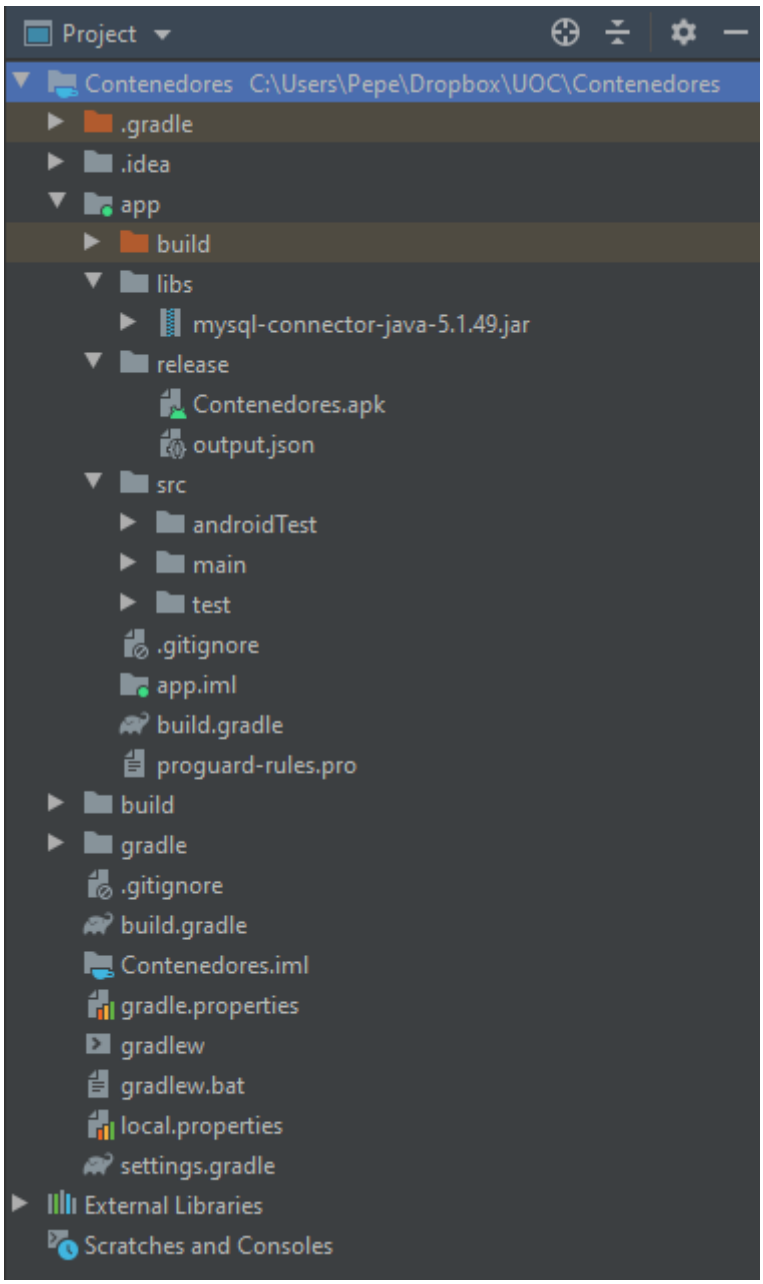


Figura 19: Árbol completo del proyecto Contenedores

Libs

En este directorio es en el que se encuentran las librerías externas que se precisen. Aparece el fichero **mysql-connector-java-5.1.49.jar** que es necesario para establecer la conexión entre la aplicación y la base de datos remota MySQL. Sin él, no se ejecutaría correctamente esta funcionalidad.

Release

Cuando Android Studio genera el apk (archivo empaquetado e instalable de la aplicación) lo ubica aquí.

Src

Es el directorio donde se alojan los códigos fuente o *source*. Todas las clases de Java están ubicadas por sus distintas subcarpetas, y como se explica en la tabla 22, el subdirectorio **androidTest** contiene las pruebas instrumentadas (las de interfaz de usuario). El fichero MainActivityTest.java almacena varios de estos test y necesita para su correcta ejecución la clase ToastMatcher.java. Por otra parte, en el subdirectorio **test** se guardan las pruebas unitarias, en este caso en el archivo UnitTest.java, formado por varias de ellas. En la sección 3.4. se describirán más en profundidad los test realizados, tanto manuales como automáticos.

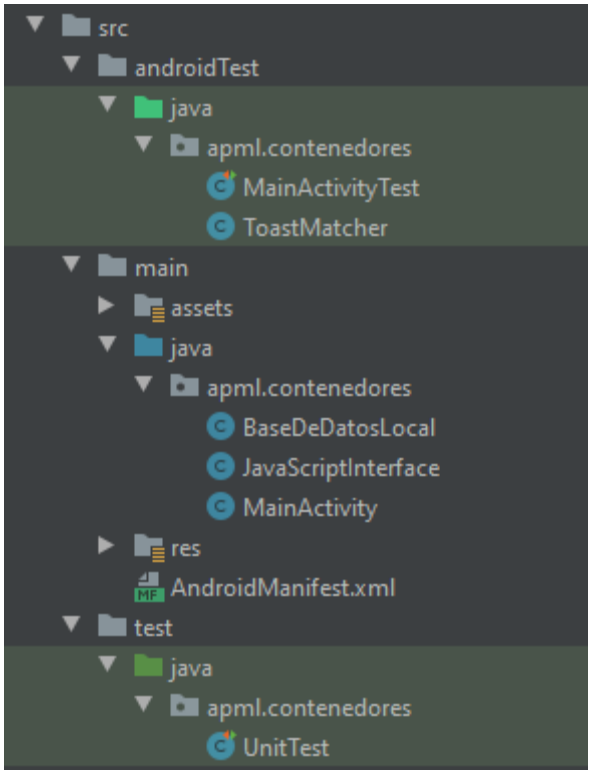


Figura 20: Directorio src con sus distintas clases Java

Main

Dentro de **src**, es necesario mencionar especialmente este directorio, que por un lado contiene el archivo **AndroidManifest.xml** y por otro, sostiene tres subdirectorios muy importantes: **assets, java** y **res**.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="apml.contenedores">

    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Contenedores"
```

```
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Figura 21: Contenido de AndroidManifest.xml

AndroidManifest.xml es sensible, pues contiene configuraciones y permisos que se aplican a la *app*. Aquí concede autorización para que la utilidad salga a Internet.

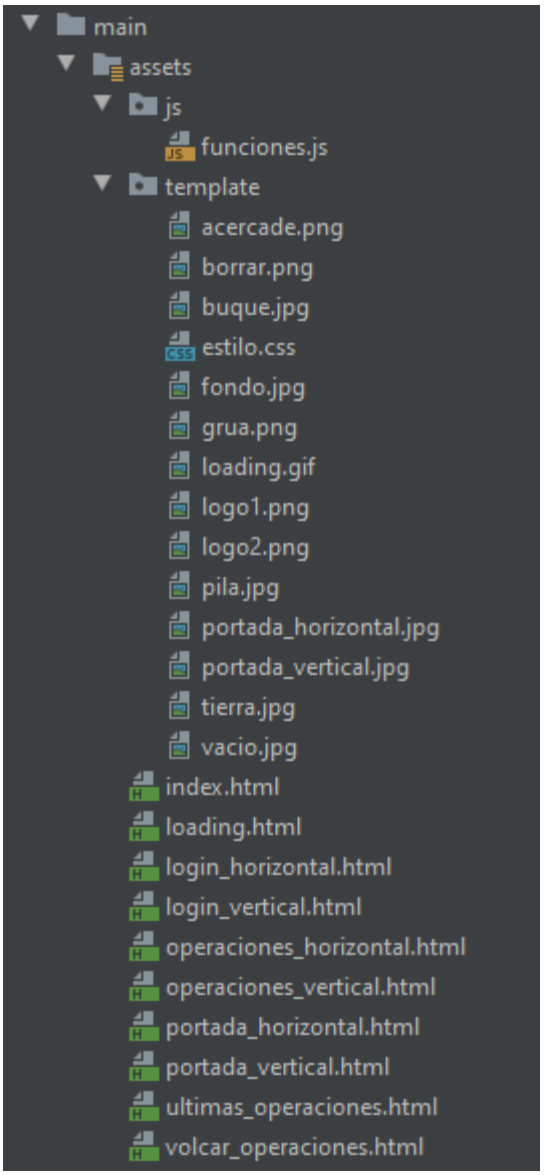


Figura 22: Directorio main, subdirectorios y contenido

Main - assets

Al ser una aplicación híbrida, trabaja con páginas HTML, que se alojan en esta ubicación. El subdirectorio **js** guarda en su interior el archivo JavaScript **funciones.js**, el cual, a la hora de interactuar con los distintos ficheros de hipertexto que emplea la herramienta, ejecutará diferentes acciones. En **template**, aparece **estilo.css** (la hoja de estilos que se le aplica a los distintos HTML) y es un repositorio de imágenes necesarias para la *app*.

Main - java

Contiene las tres clases fundamentales en las que se basa la utilidad:

- **BaseDeDatosLocal.java**, para crear objetos SQLite
- JavaScrptInterface.**java**, contiene la lógica de la aplicación, es una interfaz que permite interactuar entre Java, HTML y JavaScript.
- **MainActivity.java**, es la clase principal con la que inicia la *app*.

Main - res

Almacena recursos (*resources*) que usa la aplicación como iconos para diferentes resoluciones, imágenes, archivos de idiomas, etc.

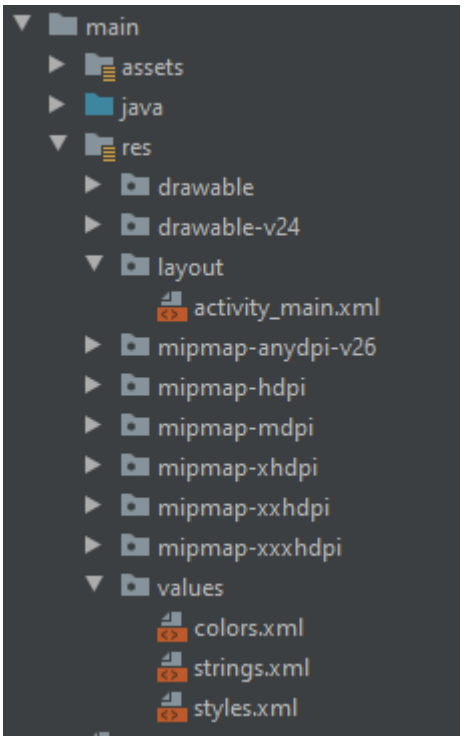


Figura 23: Directorio res, subdirectorios y contenido

El subdirectorio **layout** dispone del archivo **activity_main.xml**, que configura todo lo relacionado con las interfaces gráficas, así como la vista o vistas (*webviews)* con las que trabaja cualquier utilidad. La aplicación desarrollada en este TFG emplea una única *webview*. Por último, en **values**, el fichero strings.xml tiene un parámetro que define el nombre de la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <WebView
        android:id="@+id/webview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</RelativeLayout>
```

Figura 24: Contenido de activity_main.xml

3.3.2. Diseño y codificación

Para empezar es necesario realizar una aclaración muy importante: en este TFG se conocía con anterioridad el *hardware* y el *software* en el que se iba a emplear la utilidad, una **tablet de 10,1 pulgadas** Samsung Galaxy Tab A (2016), **con resolución de 1920 x 1200 píxeles y sistema operativo Android Nougat (versión 7.0). Este detalle es fundamental, pues todo el desarrollo se ha realizado para que funcione perfectamente en estas condiciones**. Seguidamente se detallan algunos aspectos reseñables de la implementación, es decir, el trabajo de codificación de las clases, bases de datos, algoritmos, procesos e interfaces del diseño establecidos.

La aplicación comienza su ejecución a través de la clase MainActivity.java, que es la que lanza todo el proceso.

```
package apml.contenedores;

import android.app.Activity;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

/**
 * Clase principal que ejecuta el WebView de la aplicación
 * @author José Ramón Cabanillas García
 * @version 05/06/2020
```

```
 */
public class MainActivity extends Activity {

    private WebView contenedores;
    private String id_trabajador_sesion;
    private SQLiteDatabase db_contenedores;

    /**
     * Instancia la base de datos local y la vista web
     * @param savedInstanceState Instancia
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); // Asociamos el contenido de
la vista a activity_main.xml
        BaseDeDatosLocal db = new BaseDeDatosLocal(this, "db_contenedores",
null, 1); // Creamos el objeto que contendrá la base de datos local
        db_contenedores = db.getWritableDatabase(); // Establecemos como
editable la base de datos y la inicializamos
        db.insercion_inicial(db_contenedores); // Realizamos en db la
inserción inicial de datos necesarios para trabajar con la base
        contenedores = (WebView) findViewById(R.id.webview); // Asignamos la
vista a una variable WebView
        contenedores.setWebViewClient(new WebViewClient()); // Para que los
enlaces se abran en el navegador de la propia app
        WebSettings configuracion = contenedores.getSettings(); // Asignamos
a una variable la configuración de la vista
        configuracion.setJavaScriptEnabled(true); // En esta configuración,
permitimos que se ejecute JavaScript
        contenedores.addJavascriptInterface(new JavaScriptInterface(this,
db_contenedores, id_trabajador_sesion, contenedores), "Android"); // Añadimos
a nuestra vista un objeto JavaScript llamado "Android" para usar sus
funciones
        contenedores.loadUrl("file:///android_asset/index.html"); // Cargamos
la página inicial en la vista
    }
}
```

Figura 25: Clase MainActivity.java

MainActivity.java

Inicialmente, en esta clase se desarrolló toda la herramienta y el código que la compone. Esta decisión fue escogida porque tal y como se recogió en las entrevistas de usuario, los trabajadores necesitan una utilidad clara, sencilla y sobre todo, lo más rápida posible a la hora de ejecutar operaciones; y la verdad es que resultó muy eficaz. No obstante, hay que reconocer que esta solución conllevó desventajas, como por ejemplo la dificultad en la lectura del código, aparte de contradecir ciertas normas de programación. Tras indicaciones del profesor colaborador, además de por intentar seguir una mejor praxis de implementación, se rehízo el código y se dividió en tres clases:

1. **MainActivity**, que continua como la principal (figura 25).

2. **BaseDeDatosLocal**, que es la encargada de crear objetos SQLite.

3. **JavaScriptInterface**, que es una clase de Java que sirve de interfaz y permite que desde HTML se empleen funciones JavaScript que ejecutan métodos Java implementados aquí.

MainActivity crea la instancia de la *app* (actividad) y le va dando forma: le asocia el fichero activity_main.xml a una vista, crea la base de datos local SQLite y la inicializa, asocia la vista a un *webview* concreto y le añade una interfaz JavaScriptInterface que llama "Android" (para poder invocar desde HTML las funciones JavaScript) y por último, carga el fichero index.html. Esto es lo que realiza MainActivity, que delega en el archivo de hipertexto los siguientes pasos y la interactividad con el usuario.

BaseDeDatosLocal.java

Se limita tan solo a crear objetos SQLite y tiene un método para inicializarlos. Como se ha explicado con anterioridad, la aplicación Contenedores es *ad hoc* y será usada por los mismos cuatro operarios que la tendrán instalada en dos tabletas para su actividad. Se ha decidido este diseño simple para acelerar en la medida de lo posible todos los procesos, en este caso los de *backend*. **OnCreate** se ejecuta tan solo la primera vez que se instala la *app* en un dispositivo y crea las tablas y su estructura. **OnUpdate** se lanza si la *database* está creada con anterioridad y su número de versión almacenado es inferior al nuevo (este versionado es enviado por parámetros). Por último, el método **inserción_inicial** recibe una base de datos SQLite como parámetro y la inicializa directamente con los trabajadores y los consignatarios. Así se evitan pérdidas de tiempo y errores innecesarios. Si en un futuro cambian los actores involucrados -ya sean conductores o empresas consignatarias- tan solo hay que reflejarlo en este método.

JavaScrptInterface.java

Los métodos aquí implementados son fundamentales para la interactividad del usuario con las páginas HTML que muestra la aplicación. Tiene varios, que consiguen realizar múltiples tareas como algunas de las siguientes:

- Mostrar mensajes de tipo *Toast*.
- Publicar mensajes de alerta de tipo *AlertDialog*.
- Crear el listado desplegable de los trabajadores para hacer *login*.
- Elaborar el listado desplegable para seleccionar consignatario.
- Comprobar credenciales.

- Calcular dígito de control de una matrícula (figura 26).
- Insertar operación (figura 27).
- Eliminar operación.
- Finalizar sesión.
- Enviar datos al servidor remoto (figura 28).
- Etc.

Existe un detalle importante que quisiera reseñar, y es que las operaciones y consultas (los "SELECT" con ***rawQuery*** y los "INSERT" y "DELETE" con ***execSQL***) realizadas con la base de datos se escriben directamente, es decir, no se han creado métodos específicos en la clase BaseDeDatosLocal para ello, sino que se construye la cadena SQL con la ayuda de parámetros. La razón es por efectividad y velocidad de ejecución, pues se evitan más llamadas a funciones, paso de variables, referencias, etc. que sobrecargan memoria y añaden latencia al proceso. Pienso que actuar de esta forma es más rápido y eficaz.

A continuación aparece el método para calcular el dígito de control de una matrícula, que emplea todos los valores de esta para realizarlo.

```
/**
 * Calcula el dígito de control de la matrícula del contenedor
 * @param matricula_texto  Los 3 primeros caracteres de la matrícula
 * @param matricula_u      El carácter U de la matrícula
 * @param matricula_numero Los 6 primeros dígitos de la matrícula
 * @see "http://mundopuerto.blogspot.com.es/2009/09/calculo-del-digito-de-
control-en-la.html"
 * @see "https://www.bic-code.org/calculate-the-check-digit-online.html"
 * @return El dígito de control
 */
@JavascriptInterface
public int matricula_calcula_digito_control(String matricula_texto, String
matricula_u, String matricula_numero) {
    matricula_texto = matricula_texto.toUpperCase();
    Map<Character, Integer> valores = new HashMap<>(); // Creamos HashMap con
los valores recogidos de http://mundopuerto.blogspot.com/2009/09/calculo-del-
digito-de-control-en-la.html
    valores.put('A', 10); valores.put('B', 12); valores.put('C', 13);
valores.put('D', 14); valores.put('E', 15); valores.put('F', 16);
valores.put('G', 17);
    valores.put('H', 18); valores.put('I', 19); valores.put('J', 20);
valores.put('K', 21); valores.put('L', 23); valores.put('M', 24);
valores.put('N', 25);
    valores.put('O', 26); valores.put('P', 27); valores.put('Q', 28);
valores.put('R', 29); valores.put('S', 30); valores.put('T', 31);
valores.put('U', 32);
    valores.put('V', 34); valores.put('W', 35); valores.put('X', 36);
valores.put('Y', 37); valores.put('Z', 38);
    int valor0 = (int) (valores.get(matricula_texto.charAt(0)) *
Math.pow(2,0));
    int valor1 = (int) (valores.get(matricula_texto.charAt(1)) *
```

```
Math.pow(2,1));
    int valor2 = (int) (valores.get(matricula_texto.charAt(2)) *
Math.pow(2,2));
    int valor3 = (int) (valores.get(matricula_u.charAt(0)) * Math.pow(2,3));
    int valor4 = (int)
(Integer.parseInt(String.valueOf(matricula_numero.charAt(0))) * Math.pow(2,
4));
    int valor5 = (int)
(Integer.parseInt(String.valueOf(matricula_numero.charAt(1))) * Math.pow(2,
5));
    int valor6 = (int)
(Integer.parseInt(String.valueOf(matricula_numero.charAt(2))) * Math.pow(2,
6));
    int valor7 = (int)
(Integer.parseInt(String.valueOf(matricula_numero.charAt(3))) * Math.pow(2,
7));
    int valor8 = (int)
(Integer.parseInt(String.valueOf(matricula_numero.charAt(4))) * Math.pow(2,
8));
    int valor9 = (int)
(Integer.parseInt(String.valueOf(matricula_numero.charAt(5))) * Math.pow(2,
9));
    int suma = valor0 + valor1 + valor2 + valor3 + valor4 + valor5 + valor6 +
valor7 + valor8 + valor9;
    int modulo = suma % 11;
    char control =
(Integer.toString(modulo).charAt(Integer.toString(modulo).length()-1));
    control = (char) Integer.parseInt(String.valueOf(control));
    return control;
}
```

Figura 26: Método que calcula el dígito de control de una matrícula

Este método esta testado de forma automática como se comprobará en el punto 3.4. Pruebas.

Para insertar una operación en la base de datos local se usa el método inserta_operacion:

```
/**
 * Inserta una operación
 * @param matricula_texto   Los 3 primeros caracteres de la matrícula
 * @param matricula_u       El carácter U de la matrícula
 * @param matricula_numero  Los 6 primeros dígitos de la matrícula
 * @param matricula_control El dígito de control de la matrícula
 * @param origen            El origen del contenedor
 * @param destino           El destino del contenedor
 * @throws ClassNotFoundException
 * @throws SQLException
 */
@JavascriptInterface
public void inserta_operacion(String id_consignatario, String
matricula_texto, String matricula_u, String matricula_numero, String
matricula_control, String vacio, String origen, String destino) throws
ClassNotFoundException, SQLException {
    String matricula = matricula_texto.toUpperCase() + matricula_u +
matricula_numero + matricula_control;
    Date = new Date();
```

```
    String fecha = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(date);
    db_contenedores.execSQL("INSERT INTO operacion (id_trabajador,
id_consignatario, matricula, vacio, origen, destino, fecha) VALUES ('" +
id_trabajador_sesion + "', '" + id_consignatario + "', '" + matricula + "',
'" + vacio + "', '" + origen + "', '" + destino + "', '" + fecha + "')");
}
```

Figura 27: Método para insertar operaciones en la base de datos SQLite

Ejecuta la inserción de un movimiento realizado con un contenedor.

La siguiente figura muestra el código para, una vez finalizada la sesión de trabajo, enviar las operaciones llevadas a cabo al servidor remoto. En caso de no existir conexión por falta de red, cobertura, etc. la información queda almacenada y a salvo en la base de datos local, que en la siguiente sesión tratará de mandar las operaciones. Mientras que no haya cobertura, los registros quedan en local, de ahí que la aplicación pueda actuar perfectamente en modo *offline*.

```
    /**
     * Conecta a la base de datos de APML y realiza un volcado de la base de
datos local
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    @JavascriptInterface
    public void volcar_operaciones() throws ClassNotFoundException,
SQLException {
        Cursor resultado = db_contenedores.rawQuery("SELECT * FROM operacion
WHERE id_trabajador = '" + id_trabajador_sesion + "'", null);
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection conexion =
DriverManager.getConnection("jdbc:mysql://192.168.20.44/contenedorespruebas",
"contenedores", "c0nT@1n_er");
            Statement insercion = conexion.createStatement(); // Establecemos
conexión con los datos proporcionados
            while (resultado.moveToNext()) {
                insercion.executeUpdate("INSERT INTO operacion
(id_trabajador, id_consignatario, matricula, vacio, origen, destino, fecha)
VALUES ('" + resultado.getString(1) + "', '" + resultado.getString(2) + "',
'" + resultado.getString(3) + "', '" + resultado.getString(4) + "', '" +
resultado.getString(5) + "', '" + resultado.getString(6) + "', '" +
resultado.getString(7) + "')");
                db_contenedores.execSQL("DELETE FROM operacion WHERE
id_operacion = '" + resultado.getString(0) + "'"); // Una vez enviadas
correctamente las operaciones, se vacía la tabla que las almacena
            }
            conexion.close();
        } catch (ClassNotFoundException | SQLException e) {
            muestra_alerta("AVISO - No hay conexión", "Posiblemente la tablet
no tenga conexión a Internet\n\nLas operaciones permanecerán cargadas en la
tablet hasta que la conexión se restablezca y se finalice sesión nuevamente",
"ACEPTAR");
            e.printStackTrace();
        }
        contenedores.post(new Runnable() { // Tras finalizar el proceso de
```

```
volcado de datos, cargamos en la vista la página inicial
            @Override
            public void run() {
                contenedores.loadUrl("file:///android_asset/index.html");
            }
        });
    }
```

Figura 28: Método que vuelca las operaciones en el servidor remoto

Al crear la cadena de conexión, la figura 28 muestra el servidor de pruebas empleado para los test realizados (jdbc:mysql://192.168.20.44/contenedorespruebas). En la versión de producción de la *app*, esta conexión se hace a un DNS que se ha configurado en los sistemas de la Autoridad Portuaria de Melilla. Además, para securizar algo más la conexión, se ha empleado un puerto externo de valor elevado que ha sido "nateado" en los enrutadores de la organización, que internamente lo traducen al 3306, el habitual para los sistemas MySQL.

3.4. Pruebas

Probar cualquier *app* es una parte integral del proceso de su desarrollo. Al realizar pruebas de manera coherente, se puede verificar la corrección, el comportamiento funcional y la usabilidad antes de lanzarla y presentarla al público en general. Es necesario saber si el código funciona, por lo que en el proceso de desarrollo, se deben emplear herramientas que verifiquen la correcta ejecución, además de cumplir con la calidad deseada para ese producto. [13]

Los test que se han realizado sobre la aplicación han sido probados en el emulador de Android Studio a través de AVD *(Android Virtual Device)*, que es un dispositivo virtual creado con las características necesarias para simular el hardware y software sobre el que se ejecutará nuestra aplicación: tablet de 10,1 pulgadas, con resolución de 1920 x 1200 píxeles y sistema operativo Android Nougat (versión 7.0, API 24).

Se han ideado y llevado a cabo varias pruebas **unitarias e instrumentadas (de interfaz de usuario, UI)**, algunas acordes con los requisitos funcionales especificados en el apartado **1.2.1.** Se ha empleado **JUnit4** como librería para los test, además de **Espresso** para las pruebas de interfaz de usuario.

3.1. *Frameworks*

JUnit4 [9] [10]

Es un conjunto de bibliotecas formadas por varias clases que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. El *framework* JUnit permite ejecutar clases de Java de manera controlada, para poder evaluar si el funcionamiento de los métodos de

una clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit retornará un fallo en el método correspondiente.

Espresso [11] [12]

Es un *framework* para Android desarrollado por Google que permite crear pruebas de interfaz de usuario (UI o *user interface*) para simular las interacciones de las personas con una aplicación Android, y así evitar que estas se encuentren con resultados inesperados. Por esta razón, es recomendable la creación de un entorno de pruebas vinculadas a la UI con el fin de asegurarnos que la aplicación está funcionando correctamente. Espresso permite realizar test tanto en dispositivos físicos como virtuales. Su principal ventaja es que nos permite la sincronización automática de las acciones de las pruebas con la interfaz de usuario de nuestra aplicación. [14]

Una vez detallados los *frameworks*, el siguiente paso es describir algunos de los test, tanto manuales como automáticos, que ha superado la aplicación. La nomenclatura empleada en las distintas pruebas es PRF-XX donde PRF significa prueba del requisito funcional y XX es un identificador de dos dígitos.

PRF-01 relacionado con RF1: Autenticación exitosa.

- ✓ Se selecciona usuario de la lista desplegable y se introduce su contraseña. Si se hace correctamente, se accede a la utilidad.

PRF-02 relacionado con RF1: Autenticación fallida.

- ✓ Si no se selecciona usuario de la lista, el sistema avisa de ello. Si se elige usuario pero no se introduce contraseña, también se anuncia. Si se introduce contraseña errónea, la aplicación lo indica.

PRF-03 relacionado con RF2: Identificación de un contenedor mediante la validación de su matrícula.

Una matrícula de contenedor está formada por una secuencia alfanumérica de cuatro letras y siete números que se definen así [8]:
Código del propietario: Son las tres primeras letras e identifican al propietario del contenedor.
Identificador de equipo (categoría): Es la cuarta letra, que resulta ser la U en el caso de los contenedores marítimos, por tanto, para la *app* siempre tendrá este valor.
Número de serie: Seis cifras numéricas que elige el propietario al azar, aunque este no puede repetir ese valor.

Dígito de control: Es un único dígito que se calcula en base al resto de valores y permite verificar que una secuencia es correcta.

- ✓ Si se inserta correctamente una matrícula (las tres primeras letras y los seis dígitos) el sistema calcula automáticamente el número de control (ubicado en la última posición).

```
@Test
/**
 * Comprueba que el dígito de control calculado de una matrícula coincide con
el original
 * Supera el test con una matrícula real de la que se conoce su dígito de
control
 */
public void matricula_calcula_digito_control_test() {
    int resultado = jsi.matricula_calcula_digito_control("TCL", "U",
"672226");
    assertEquals(7, resultado);
}
```

Figura 29: Código del test del requisito funcional PRF-03

PRF-04 relacionado con RF2: Identificación fallida de un contenedor.

- ✓ Si no se introduce o se hace de manera errónea, aparece un aviso indicándolo.

PRF-05 relacionado con RF3: Selección del origen de un contenedor.

- ✓ Si no se selecciona, muestra un aviso indicándolo.

PRF-06 relacionado con RF4: Selección del destino de un contenedor.

- ✓ Si no se selecciona, muestra un aviso indicándolo.

PRF-07 relacionado con RF5: Marcador que identifica si un contenedor está vacío o lleva carga.

- ✓ Si está marcado, identifica al contenedor como vacío.
- ✓ Si no está marcado, identifica al contenedor con carga.

PRF-08 relacionado con RF6: Registro de los distintos movimientos de contenedores.

- ✓ Aparecen reflejados los cinco últimos movimientos insertados, ordenados cronológicamente de más reciente a más antiguo.

PRF-09 relacionado con RF7: Borrado de datos introducidos. Confirmación.

- ✓ Aparece advertencia de eliminación que muestra los valores del registro indicado, queda a la espera de confirmación o cancelación. Al confirmar, se suprime el registro y actualiza los datos mostrados.

PRF-10 relacionado con RF7: Borrado de datos introducidos. Cancelación.

- ✓ Si se cancela, muestra mensaje breve (tipo *Toast*) indicándolo.

```
@Test
/**
 * Comprueba el aparece correctamente un mensaje Toast específico
 * Esta prueba se pasa si la orientación es vertical. Falla en caso contrario
 */
public void toast_muestra_texto_especifico_test() {
    onView(withText("Cambiando a orientación vertical..."))
            .inRoot(new ToastMatcher())
            .check(matches(isDisplayed()));
}
```

Figura 30: Código del test del requisito funcional PRF-10

Seguidamente se añaden más capturas del código empleado para realizar algunos de los diferentes test:

```
/**
 * Al mostrar la portada,hay que hacer click para pasar a la página de login
 */
@Test
public void con_un_click_muestra_pagina_login() {
    onView(withId(R.id.webview)).perform(click());
}
```

Figura 31: Código del test que comprueba que se hace una pulsación (click) en la portada y se pasa a página de *login*

```
/**
 * Comprueba que se usa el contexto correcto de la aplicación
 */
@Test
public void usa_contexto_app_test() {
    Context contexto_app = mActivityTestRule.getActivity().getBaseContext();
    assertEquals("apml.contenedores", contexto_app.getPackageName());
}
```

Figura 32: Código del test que comprueba que las acciones se realizan en el contexto correcto

```
/**
 * Comprueba que se crea un objeto BaseDeDatosLocal
 */
@Test
public void base_datos_local_creada_test() {
    BaseDeDatosLocal bd = new BaseDeDatosLocal(c, "bd_test", null, 1);
```

```
    assertNotNull(bd);
}
```

Figura 33: Código del test que verifica la correcta creación de una base de datos

```
/**
 * Crea correctamente nuestra vista identificada en MainActivity como
"webview"
 */
@Test
public void webview_ejecutado_correctamente() {
    MainActivity mActivity = mActivityTestRule.getActivity();
    WebView vista = mActivity.findViewById(R.id.webview);
    assertNotNull(vista);
}
```

Figura 34: Código del test que verifica la correcta creación de la vista (webview) que emplea la *app*

A continuación, se adjuntan los resultados de los distintos test automáticos realizados a la aplicación, **tanto unitarios** (del que se han ejecutado tres pruebas) **como instrumentados** (del que se han realizado cuatro pruebas):

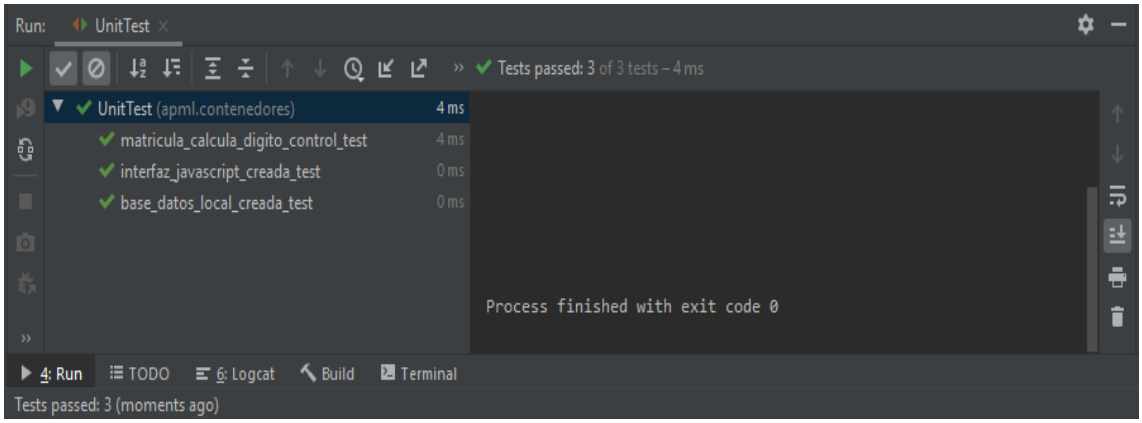


Figura 35: Resultados de los test unitarios o de unidad local (JUnit)

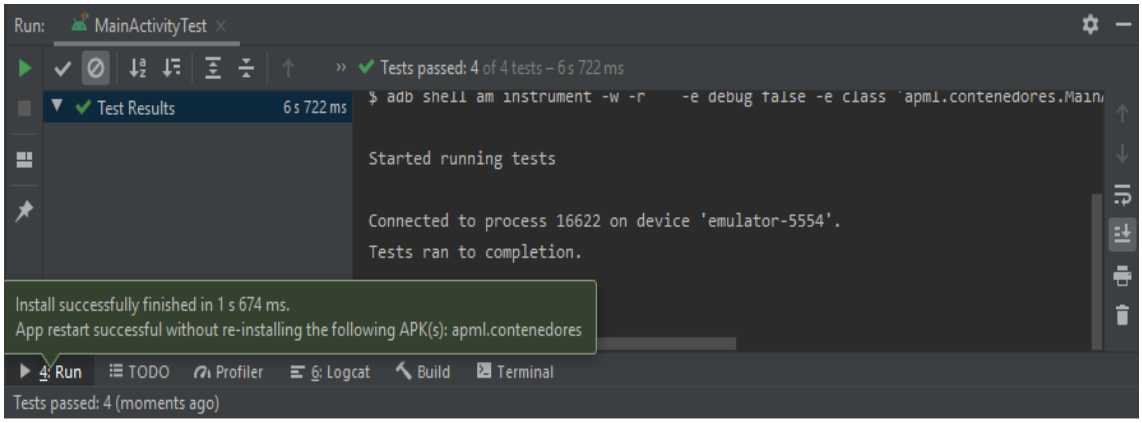


Figura 36: Resultados de los test instrumentados de interfaz de usuario (Espresso)

Como se puede apreciar en las dos últimas figuras, todas las pruebas (siete en total) han concluido satisfactoriamente.

Tal y como se ha indicado en el apartado anterior, había quedado pendiente un objetivo funcional que consiste en enviar la información almacenada en la base de datos SQLite de la aplicación al servidor remoto. Esta contrariedad se corrigió y está lista y perfectamente operativa para la entrega final.

Por último, se adjuntan imágenes de algunas pruebas unitarias efectuadas:

- PRF-02:

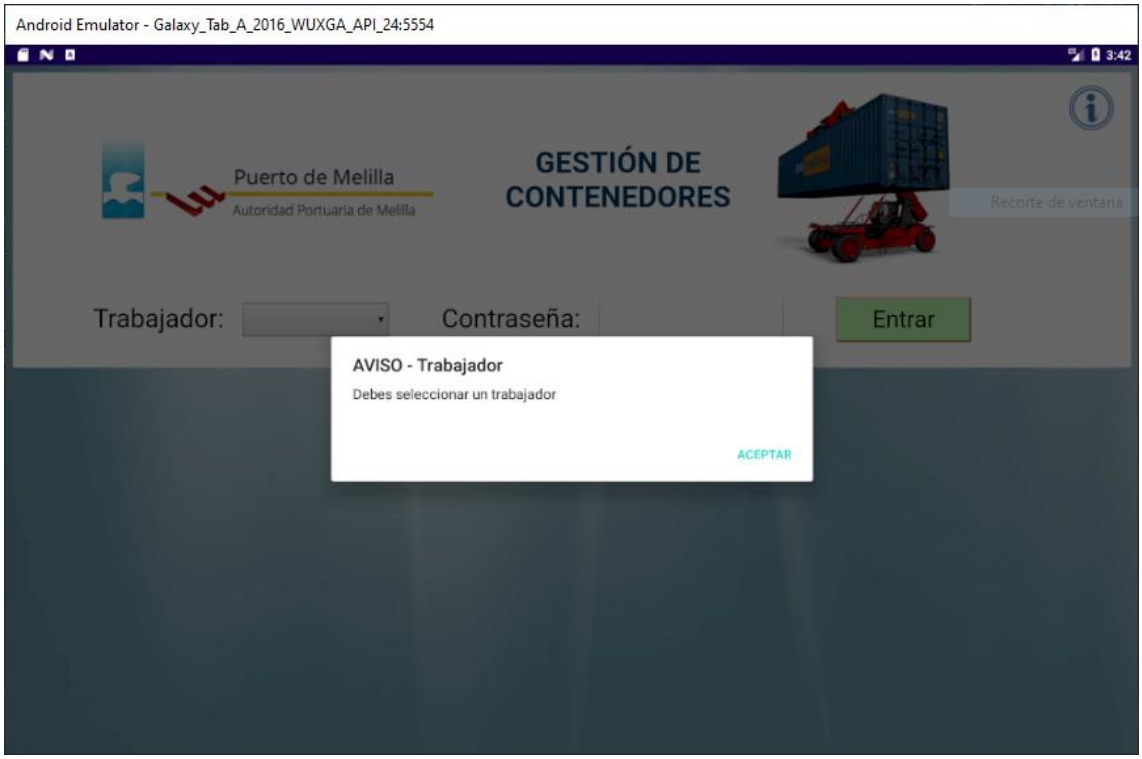


Figura 37: Prueba de requisito funcional 02

- PRF-04:

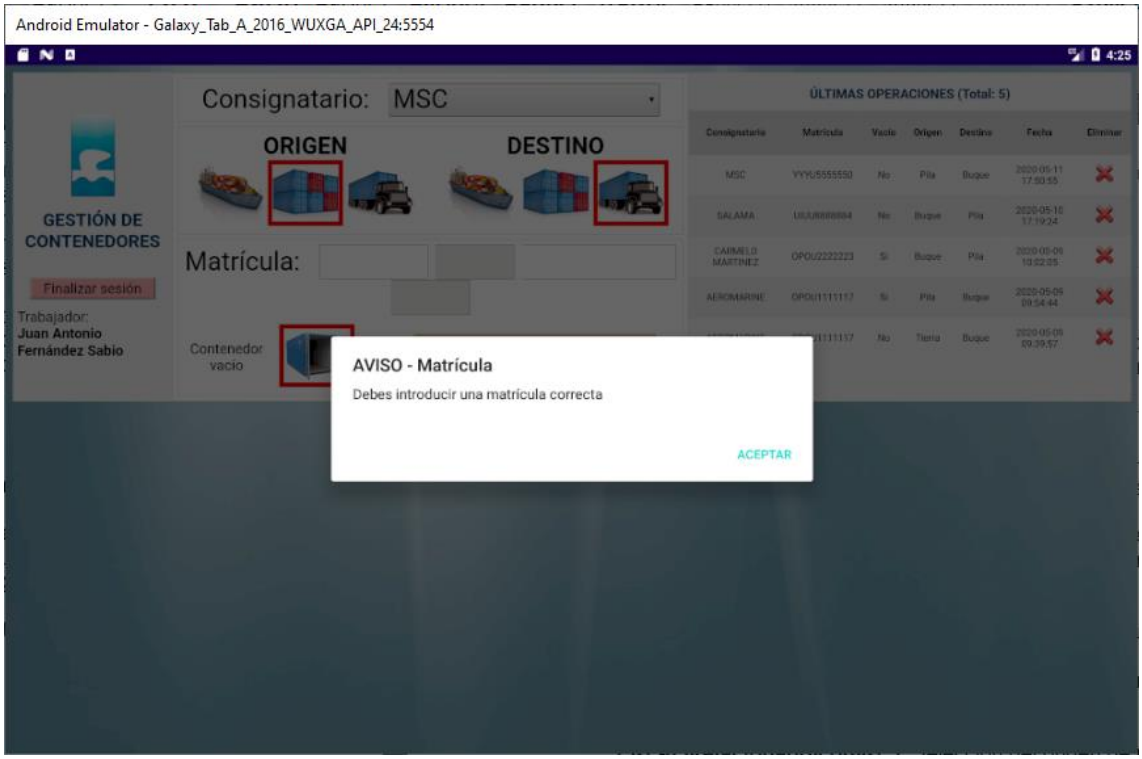


Figura 38: Prueba de requisito funcional 04

- PRF-09

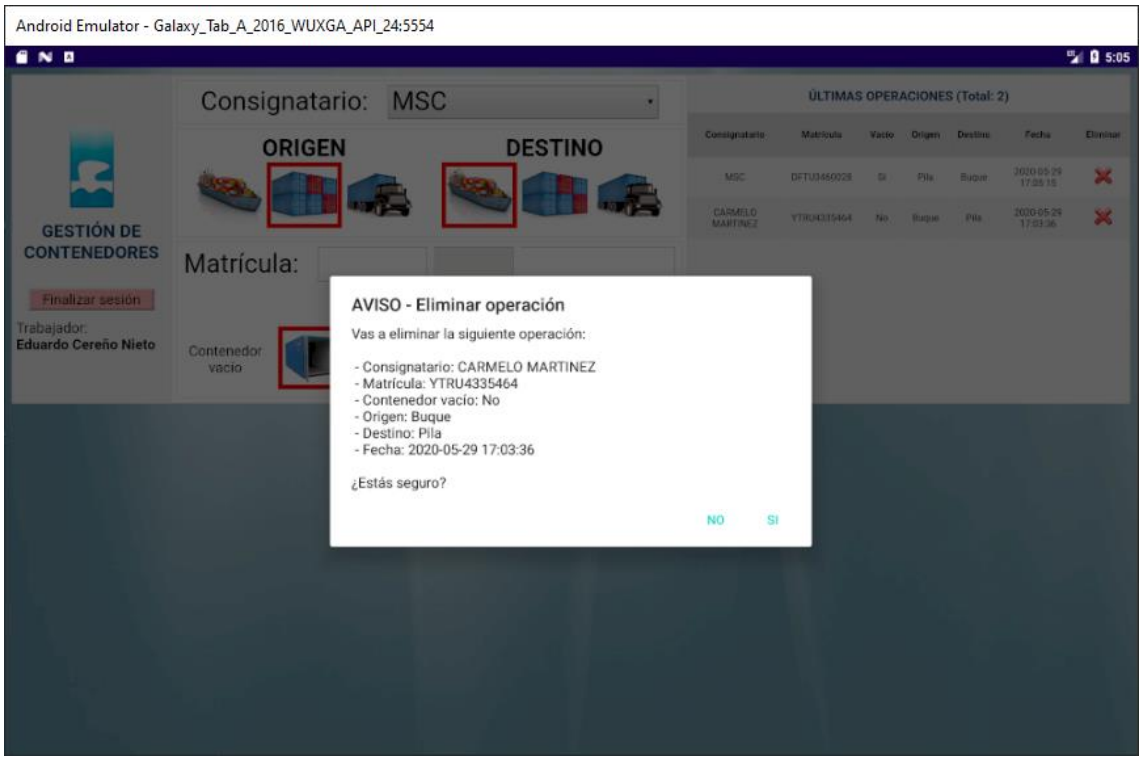


Figura 39: Prueba de requisito funcional 09

- PRF-10

Figura 40: Prueba de requisito funcional 10

4. Conclusiones

Con toda sinceridad, este semestre comenzó muy bien, con buenas expectativas, pues se supone que era el último que iba a cursar, pero ha terminado de forma bastante extraña. Entre medias apareció en el quehacer diario una desgraciada pandemia que, guste o no, ha afectado de manera determinante la vida de todas las personas, sin excepción. Esta circunstancia, que sigue vigente en la actualidad, ha marcado también el Trabajo de Fin de Grado. ¿Por qué?

El estar recluido en casa -que de inicio parece una opción ideal para avanzar- ha resultado un enorme quebradero de cabeza, pues a la vida estudiantil habitual, con dos asignaturas además de este TFG, se ha añadido el teletrabajo: laboralmente surgen todo tipo de problemas, de forma continua y a deshoras que se agregan a la faena diaria. Prácticamente ha sido una atención al usuario 24/7. Además, se ha sumado la vida familiar, que obliga a atender las tareas de casa y de los hijos de forma aún más intensiva. Una mezcla que ha afectado directamente al desarrollo de este proyecto.

Sin embargo, no todo ha sido negativo pues he aprendido que tengo capacidad de sacrificio, dedicación y esfuerzo, de dormir poco y trabajar mucho para superar contrariedades. Gracias a estas cualidades y a la inestimable e imprescindible ayuda y abnegación de mi mujer, sin la cual seguro que no hubiera concluido a tiempo, se consiguió remediar una pequeña desviación en los plazos, que afortunadamente pudo ser corregida y se cumplió con el calendario previsto.

Una vez finalizado el desarrollo del TFG, si se da un vistazo atrás, soy capaz de ver en perspectiva la evolución, los objetivos conseguidos y los conocimientos adquiridos. Este proyecto ha terminado de forma satisfactoria, aunque siempre es mejorable.

Hace algunos meses, al proponer el trabajo, se fijaron unos hitos que marcaban el que iba a ser un proyecto interesante, y como resultado, una herramienta útil para la APML. No obstante, estos hitos tenían una complejidad desconocida para mí en cuanto a su desarrollo. Nunca había trabajado con Android, ni programado para ningún dispositivo móvil, y siendo realista, los plazos para desarrollar esta aplicación eran, como mínimo, inciertos. Durante este periodo de actividad, a pesar del maldito COVID-19, se ha seguido bastante bien la planificación inicial, aunque en alguna fase sí que se han tenido que añadir bastantes horas de más.

Te das cuenta de que un proyecto sufre multitud de contrariedades, que afectan directa o indirectamente a las tareas y plazos y que necesitan tu atención y dedicación plena. Surgen dudas en multitud de situaciones y se intentan tomar las mejores decisiones, aunque inevitablemente, en múltiples ocasiones te equivocas. En definitiva, se ponen a prueba muchos de los conocimientos adquiridos a lo largo del Grado y otros

nuevos que sumas. Todo ello ha reportado un gran avance personal, pues asumes otros enfoques que te van enriqueciendo; es más, visualizas situaciones perfectamente plausibles en un entorno de trabajo real (que en alguna circunstancia ya he conocido), y que han logrado que atesore una mayor y mejor experiencia.

Estoy satisfecho con el TFG realizado, pues cumple de forma espléndida con el cometido para el que fue diseñado. Es más, se ha puesto en producción y debo decir que funciona perfectamente.

Por último, quisiera agradecer a los profesores que me han ayudado en este TFG, en especial a David Escuer, siempre pendiente y ágil en su apoyo. Gracias.

4.1. Líneas de trabajo futuro

Tras la puesta en producción de la aplicación Contenedores, quiero esperar unos días para hacer un pequeño cuestionario a los trabajadores que emplean la herramienta, recabar sus opiniones, críticas y aportaciones para analizarlas y así poder actuar en consecuencia.

No obstante, tengo en mente un par de circunstancias que me gustaría nombrar:

Por un lado, se debe probar la *app* en otras versiones de sistemas operativos Android (incluso testarla en otras plataformas como iOS), evaluar su comportamiento y corregir fallos en ese sentido para no depender de dicha versión. En teoría, esto ya es así, pues se trata de una *app* híbrida, basada en contenido web y que muestra páginas HTML en cualquier dispositivo y navegador, independientemente de su hardware, tamaño, resolución y sistema operativo. Esta es una de las ventajas de los diseños híbridos, pero he dicho que esto es la teoría, por lo que siempre es mejor llevarlo la práctica, cerciorarse de que no surgen problemas y que la aplicación se adapta y realiza su función correctamente.

Por otro lado, una posible mejora puede ser trabajar y almacenar otro dato adicional, concretamente la geolocalización. Cuando se ubica un contenedor en una pila, saber exactamente en qué coordenadas se ha depositado. Esto puede facilitar la labor de inspección del personal de Aduanas o la Policía Portuaria, pues de un vistazo podría posicionarte la mercancía en un plano, por ejemplo a través de Google Maps.

5. Glosario

APML: Autoridad Portuaria de Melilla.

ad hoc: Adecuado, apropiado, dispuesto especialmente para un fin.

app: Aplicación.

consignatario: Un consignatario de buques, agente marítimo o agente consignatario de buques es un intermediario independiente (persona o entidad) que actúa en nombre del propietario de un buque, ya sea naviero o armador en los puertos, y ejecuta las fases terrestres del transporte marítimo, entregando y recibiendo la carga.

desestiba: Operación contraria de la estiba es decir, el removido de la carga y su entrega al equipo de descarga para extraer de la bodega del buque la mercancía previamente estibada.

DSIC: División de Sistemas de Información y Comunicaciones de la Autoridad Portuaria de Melilla.

estiba: Son las diferentes operaciones que se realizan con las mercancías para ubicarlas correctamente en las áreas y zonas de carga.

IDE: El entorno de desarrollo integrado *(Integrated Development Environment*), es una aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo de software.

QlikView: Software desarrollado por la empresa Qlik y que se utiliza para análisis y *Business Intelligence*. Permite examinar datos y usarlos para apoyar la toma de decisiones.

TFG: Trabajo de Fin de Grado.

6. Bibliografía

[1] DEVELOPERS. "*Android Studio*" (2020) [en línea] [Consulta: 23 de febrero de 2020] <https://developer.android.com/studio>

[2] WIKIPEDIA. "*Entorno de desarrollo integrado*" (13/02/2020) [en línea] [Consulta: 03 de marzo de 2020] <https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado>

[3] INVISION. "*Digital product design, workflow & collaboration*" [en línea] [Consulta: 20 de marzo de 2020] <https://www.invisionapp.com/>

[4] JUSTINMIND. "*Free prototyping tool for web & mobile apps*" [en línea] [Consulta: 21 de marzo de 2020] <https://www.justinmind.com/>

[5] HO, D. "*Notepad++*" [en línea] [Consulta: 02 de abril de 2020] <https://notepad-plus-plus.org/>

[6] HTML-CSS-JS.COM. "*HTML - CSS - JS: The Client-Side Of The Web*" [en línea] [Consulta: 03 de abril de 2020] <https://html-css-js.com/>

[7] PIACENTINI, M. "*DB Browser for SQLite*" [en línea] [Consulta: 18 de abril de 2020] <https://sqlitebrowser.org/>

[8] BUREAU INTERNATIONAL DES CONTAINERS ET DU TRANSPORT INTERMODAL. "*BIC Codes*" [en línea] [Consulta: 15 de abril de 2020] <https://www.bic-code.org/>

[9] WIKIPEDIA. "*JUnit*" (17/01/2020) [en línea] [Consulta: 20 de mayo de 2020] < https://es.wikipedia.org/wiki/JUnit>

[10] DESARROLLADORES DE ANDROID. "*Cómo compilar pruebas de unidades locales*" (27/12/2019) [en línea] [Consulta: 22 de mayo de 2020] <https://developer.android.com/training/testing/unit-testing/local-unit-tests>

[11] DESARROLLADORES DE ANDROID. "*Espresso*" (27/12/2019) [en línea] [Consulta: 22 de mayo de 2020] <https://developer.android.com/training/testing/espresso>

[12] WIKIPEDIA. "*Espresso (framework)*" (28/09/2019) [en línea] [Consulta: 22 de mayo de 2020] <https://es.wikipedia.org/wiki/Espresso_(framework)>

[13] DESARROLLADORES DE ANDROID. "*Cómo probar tu app*" (11/05/2020) [en línea] [Consulta: 24 de mayo de 2020] <https://developer.android.com/studio/test?hl=es-419>

[14] DESARROLLADORES DE ANDROID. "*Prueba la IU para una sola app*" (27/12/2019) [en línea] [Consulta: 24 de mayo de 2020]

<https://developer.android.com/training/testing/ui-testing/espresso-testing>

Otros recursos empleados en este TFG:

- LIPIDO'S LAB. "*Interfaces de usuario HTML/CSS/Javascript ¿nuevo estándar?*" (07/05/2013) [en línea] [Consulta: 02 de abril de 2020] <https://www.sing-group.org/~lipido/blog/2013/05/07/interfaces-de-usuario-htmlcssjavascript-nuevo-estandar/>

- GEEKSFORGEEKS.ORG. "*Frontend vs Backend*" [en línea] [Consulta: 02 de abril de 2020] <https://www.geeksforgeeks.org/frontend-vs-backend/>

- BAQUERO, J. "*Desarrollo de apps con JavaScript: ¿híbrido o nativo?*" (17/07/2017) [en línea] [Consulta: 03 de abril de 2020] <https://www.arsys.es/blog/programacion/desarrollo-apps-javascript-hibrido-nativo/>

- MANEJANDO DATOS. "*Modificar COLLATE de las tablas en MySQL-MariaDB*" (17/06/2019) [en línea] [Consulta: 04 de abril de 2020] <https://www.manejandodatos.es/2019/06/modificar-collate-las-tablas-mysql-mariadb/>

- SAMSUNG. "*Galaxy Tab A (10.1 ", 4G, 2016)*" [en línea] [Consulta: 04 de abril de 2020] <https://www.samsung.com/es/tablets/galaxy-tab-a-10-1-2016-t585-32gb/SM-T585NZWEPHE/>

- STACKOVERFLOW. "*what is the difference between query() and rawQuery() in sqlite and which one is more efficient and goog [duplicate]*" (03/11/2014) [en línea] [Consulta: 08 de mayo de 2020] <https://stackoverflow.com/questions/26711050/what-is-the-difference-between-query-and-rawquery-in-sqlite-and-which-one-is/26711112>

- SGOLIVER. "*Bases de Datos en Android (III): Consultar/Recuperar registros*" (07/02/2011) [en línea] [Consulta: 10 de mayo de 2020] <https://www.sgoliver.net/blog/bases-de-datos-en-android-iii-consultarrecuperar-registros/>

- ACADEMIAANDROID.COM. "*SQLite en Android: creación y acceso base de datos e inserción de registros*" (06/10/2016) [en línea] [Consulta: 10 de mayo de 2020] <https://academiaandroid.com/sqlite-android-creacion-acceso-base-datos-insercion/>

- INVARATO, R. "*Context de Android*" (27/10/2013) [en línea] [Consulta: 15 de mayo de 2020] <https://jarroba.com/context-de-android/>

- MDN WEB DOCS. "*Screen.orientation*" (28/01/2020) [en línea] [Consulta: 30 de mayo de 2020] <https://developer.mozilla.org/en-US/docs/Web/API/Screen/orientation>

- // ELEMENTCODE. "*Make App in 5 minutes using Android webview (HTML/CSS/Javascript)*" [Consulta: 01 de abril de 2020] <https://www.youtube.com/watch?v=mNqjFhgCrlY>

- MOUREDEV BY BRAIS MOURE. "*ANDROID STUDIO: COMO Crear una APP (para Principiantes)*" [Consulta: 17 de abril de 2020] <https://www.youtube.com/watch?v=mNqjFhgCrlY>

- ARHNULD. "*Creating a very simple app with HTML, CSS & Javascript - Part 1/3 - Onsen UI*" [Consulta: 19 de abril de 2020] <https://www.youtube.com/watch?v=mNqjFhgCrlY>

- ALDOMINIUM. "*Desarrollo Android desde Principiante hasta Profesional - 20 - Cambiando el icono de la aplicación*" [Consulta: 20 de abril de 2020] <https://www.youtube.com/watch?v=mNqjFhgCrlY>

- LUJAN, J. "*Capítulo 12 - Android studio: Button y Toast*" [Consulta: 21 de abril de 2020] <https://www.youtube.com/watch?v=nak-nhxJG7E&list=PLh_neeN4BQCmX4jC9pR_DgZVJ8a4PRMca&index=12>

- DONGSU JANG. "*How to build Android App with HTML5/CSS/JavaScript*" [Consulta: 21 de abril de 2020] <https://www.youtube.com/watch?v=eJa8xie9WZs>

- HENAO, C. "*50. Como crear una Base de Datos SQLite en Android*" [Consulta: 28 de abril de 2020] <https://www.youtube.com/watch?v=9WiyqIcffe0>

- HENAO, C. "*52. Como Insertar Datos en SQLite usando sentencia SQL en Android*" [Consulta: 28 de abril de 2020] <https://www.youtube.com/watch?v=SmrpCjz-QlA>

- DESARROLLADOR CREATIVO. "*Android Studio - Cambiar icono y nombre de una app - Vídeo 5*" [Consulta: 15 de mayo de 2020] <https://www.youtube.com/watch?v=RsYGhbwDlxs>

- UNIVERSO ANDRIOD. "*Modificar Icono principal, nombre de la aplicación en Android studio*" [Consulta: 16 de mayo de 2020] <https://www.youtube.com/watch?v=pfOtcYsBzXs>

- LATINCODER. "*Pruebas Unitarias con Android Studio y Gradle - Tutorial*" [Consulta: 20 de mayo de 2020] <https://www.youtube.com/watch?v=NlpsluiZxXE>

- LEARN SHARE ANYTHING ANYONE. "*Android app development for beginners - 26 - Android - Unit test for Activity - Activity Test Rule*" [Consulta: 21 de mayo de 2020] <https://www.youtube.com/watch?v=_TR6QcRozAg>

- PRABEESH R K. "*Android WebView - Binding JavaScript code to Android code*" [Consulta: 22 de mayo de 2020] <https://www.youtube.com/watch?v=9RwJeocTgJg>

7. Anexos

7.1. Instalación de la aplicación

Se trata de un proceso sumamente sencillo, pues tan solo hay que disponer de la aplicación empaquetada (archivo **Contenedores.apk** proporcionado en los entregables, en el directorio /app/release) y enviarla al dispositivo en el que se desea instalar. Al tratarse de una herramienta *ad hoc*, ya se sabía con anterioridad a comenzar este Trabajo de Fin de Grado el *hardware* en el que se iba a emplear: **tablet de 10,1 pulgadas** Samsung Galaxy Tab A (2016) **con resolución de 1920 x 1200 píxeles y sistema operativo Android Nougat (versión 7.0). Este detalle es enormemente importante, pues todo el desarrollo se ha realizado para que funcione perfectamente en estas condiciones**. No se ha dispuesto de tiempo para probar en otros sistemas Android, por lo que se desconoce el comportamiento que pueda tener esta *app*.

7.2. Manual de usuario

MANUAL - App Contenedores

1. INTRODUCCIÓN

Esta aplicación permite llevar un control sobre los movimientos de contenedores que se realizan en la terminal de contenedores del Puerto de Melilla.

APP desarrollada por la División de Sistemas de Información y Comunicaciones de la Autoridad Portuaria de Melilla.

Versión: 05/06/2020 (Android)
Contacto: informatica@puertodemelilla.es

2. INICIO

Abrir la aplicación desde el icono:

Colocar la tablet en la orientación deseada y tocar en cualquier parte de la pantalla para continuar.

3. AUTENTICACIÓN

A. Ver información de la aplicación.
B. Seleccionar trabajador.
C. Introducir contraseña.
D. Acceder a la aplicación con las credenciales introducidas.

LISTADO DE TRABAJADORES

Nombre	Usuario	Contraseña
Antonio C. N.	acereno	acn1
Eduardo C. N.	ecereno	ecn1

Juan Antonio F. S.	jfernandez	jfs1
Juan Antonio H. C.	jhernandez	jhc1

4. OPERACIONES

A. Seleccionar consignatario.

B. Seleccionar origen del contenedor.

C. Seleccionar destino del contenedor.

D. Introducir matrícula del contenedor.

E. Seleccionar si el contenedor está vacío.

F. Insertar la operación.

G. En esta sección irán apareciendo las operaciones introducidas.

Se pueden eliminar pulsando sobre la X roja.

H. Una vez finalizado el trabajo, pulsar sobre Finalizar sesión y confirmar.