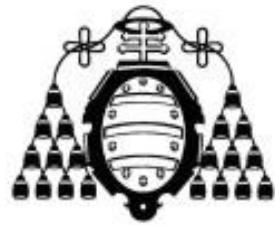


UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas.

DIRECTORES:

- **Marco Antonio García Tamargo**
- **Cristian González García**

AUTOR: Alejandro León Pereira

Agradecimientos

Se agradece a Marco Antonio García Tamargo y a Cristian González García haber hecho posible este Trabajo de Fin de Grado tutorizándolo y aprobando mi idea.

También se agradece a todas las personas que están detrás de los comandos de Linux que permitieron realizar la auditoría:

- Andrei Costin (zveriu@gmail.com) (desarrollador principal del comando mfcuk).
- Todas las personas que han colaborado con el repositorio de GitHub “nfc-tools” (Contribuidores nfc-tools, 2022), en especial las que han colaborado en la librería “libnfc” (Contribuidores libnfc, 2021), utilizada para llevar a cabo la auditoría.

Resumen

Se ha realizado una auditoría de las tarjetas MIFARE Classic, utilizadas por millones de sistemas a nivel mundial, por ejemplo, de transporte público, gimnasios, oficinas y garajes. Estas tarjetas se basan en el estándar NFC (Near Field Communication). No obstante, cabe aclarar que la auditoría no busca encontrar vulnerabilidades en la propia tecnología NFC, sino en las propias tarjetas del transporte público del Consorcio de Transportes de Asturias.

El principal motivo de la realización de la auditoría ha sido demostrar mediante pruebas empíricas, estudios y estándares, las vulnerabilidades que ya se conoce existen en las tarjetas de transporte público de Asturias, tal y como se muestra en el siguiente artículo (El Comercio, 2018).

Cabe aclarar que, si bien el Trabajo de Fin de Grado se divide en la auditoría, por una parte, y la aplicación desarrollada para Android por otra parte, la parte principal del Trabajo de Fin de Grado y a la que se le ha puesto una mayor dedicación es la auditoría, pues su realización es el principal objetivo del Trabajo de Fin de Grado.

El objetivo del proyecto es demostrar la vulnerabilidad de las tarjetas de tipo MIFARE Classic cuando se utilizan como soporte de tarjetas de transporte, y proponer alternativas a su uso. Por ejemplo, una posible alternativa al uso de tarjetas MIFARE Classic en autobuses y trenes sería el uso de una aplicación para móviles. En este trabajo se ha desarrollado una aplicación Android, que permitiría a los usuarios la compra y manejo de bonos, para utilizarlos en el transporte público desde el móvil, como alternativa al uso de tarjetas.

El proyecto se divide en dos aplicaciones:

- PassengerApp será la aplicación que llevarán los pasajeros instalada en su dispositivo Android, y que les permitirá gestionar y utilizar sus bonos del transporte público.
- VehicleApp será la aplicación que llevarán los vehículos (autobuses o trenes) en su dispositivo Android, o en el dispositivo Android de su conductor.

De esta manera, los usuarios del transporte público podrán subirse a los autobuses y trenes usando bonos mediante su dispositivo Android que soporte comunicaciones por Bluetooth. Todos los autobuses y trenes deberán tener a su vez un dispositivo Android con Bluetooth. Y los dispositivos de los pasajeros se comunicarán con el dispositivo del vehículo mediante Bluetooth. Se ha decidido utilizar Bluetooth en lugar de NFC debido a que todos los dispositivos tienen Bluetooth, pero no todos tienen NFC.

La aplicación se basa en el estándar Bluetooth Low Energy, una modalidad de Bluetooth más eficiente que el Bluetooth convencional, y que permitiría la comunicación entre los medios de transporte y los usuarios pasajeros.

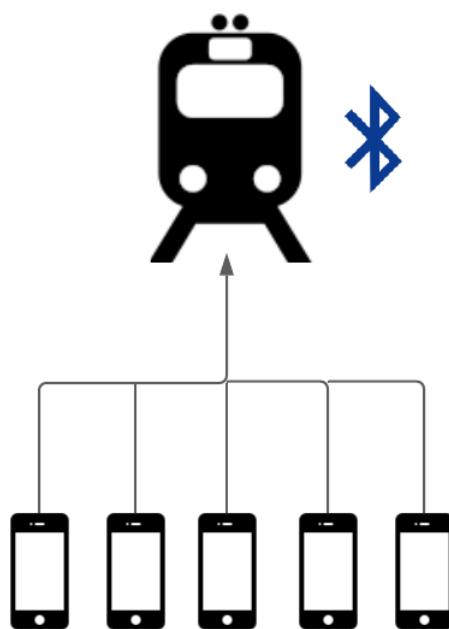


Figura 1 Esquema básico del funcionamiento de la aplicación.

En la Figura 1 se muestra un esquema del funcionamiento del sistema. Los teléfonos móviles de los usuarios se conectan por Bluetooth al vehículo.

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO.....	13
1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO	13
1.2 RESUMEN DE TODOS LOS ASPECTOS	14
1.3 LISTA DE DISTRIBUCIÓN	15
CAPÍTULO 2. INTRODUCCIÓN.....	17
2.1 JUSTIFICACIÓN DEL PROYECTO.....	17
2.2 OBJETIVOS DEL PROYECTO.....	17
2.3 ESTUDIO DE LA SITUACIÓN ACTUAL	18
2.3.1 <i>Evaluación de Alternativas.....</i>	18
CAPÍTULO 3. PLANIFICACIÓN DEL PROYECTO Y PRESUPUESTO INICIALES	25
3.1 PLANIFICACIÓN INICIAL DE LA AUDITORÍA	25
3.2 PLANIFICACIÓN INICIAL DE LAS APLICACIONES ANDROID	27
3.3 PRESUPUESTO ESTIMADO TOTAL	29
CAPÍTULO 4. ANÁLISIS	31
4.1 DEFINICIÓN DEL SISTEMA	31
4.1.1 <i>Determinación del Alcance del Sistema</i>	31
4.2 REQUISITOS DEL SISTEMA	32
4.2.1 <i>Obtención de los Requisitos del Sistema</i>	32
4.2.2 <i>Identificación de Actores del Sistema.....</i>	34
4.2.3 <i>Especificación de Casos de Uso</i>	34
CAPÍTULO 5. DISEÑO DEL SISTEMA	39
5.1 ARQUITECTURA DEL SISTEMA	39
5.1.1 <i>Diagramas de Paquetes</i>	39
5.1.2 <i>Diagramas de Despliegue</i>	42
5.2 DISEÑO DE CLASES	42
5.3 DIAGRAMAS DE INTERACCIÓN Y ESTADOS.....	48
5.3.1 <i>Diagramas comunes a las dos aplicaciones</i>	48
5.3.2 <i>Diagramas de VehicleApp</i>	50
5.3.3 <i>Diagramas de PassengerApp</i>	50
5.4 DIAGRAMAS DE ACTIVIDADES.....	53
5.5 DISEÑO DE LA BASE DE DATOS	54
5.5.1 <i>Descripción del SGBD Usado</i>	54
5.5.2 <i>Integración del SGBD en Nuestro Sistema</i>	54
5.5.3 <i>Diagrama E-R</i>	56
5.6 DISEÑO DE LA INTERFAZ	57
CAPÍTULO 6. IMPLEMENTACIÓN DEL SISTEMA	59
6.1 ESTÁNDARES Y NORMAS SEGUIDOS.....	59
6.2 LENGUAJES DE PROGRAMACIÓN	60
6.3 HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO.....	61
6.3.1 <i>Android Studio.....</i>	61
6.3.2 <i>Git.....</i>	61
6.3.3 <i>scrcpy</i>	61

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas.

6.4	CREACIÓN DEL SISTEMA.....	62
6.4.1	<i>Problemas Encontrados</i>	62
6.4.2	<i>Descripción Detallada de las Clases</i>	63
CAPÍTULO 7.	DESARROLLO DE LAS PRUEBAS	69
7.1	PRUEBAS UNITARIAS.....	69
7.2	PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA.....	75
7.3	PRUEBAS DE USABILIDAD	77
7.4	PRUEBAS DE RENDIMIENTO	81
CAPÍTULO 8.	MANUALES DEL SISTEMA	83
8.1	MANUAL DE INSTALACIÓN	83
8.2	MANUAL DE USUARIO.....	86
8.2.1	<i>Partes comunes de la aplicación</i>	86
8.2.2	<i>VehicleApp</i>	88
8.2.3	<i>PassengerApp</i>	90
8.3	MANUAL DEL PROGRAMADOR.....	104
8.3.1	<i>Manual del programador para PassengerApp</i>	104
8.3.2	<i>Manual del programador para VehicleApp</i>	107
CAPÍTULO 9.	CONCLUSIONES Y AMPLIACIONES	111
9.1	CONCLUSIONES.....	111
9.2	AMPLIACIONES	111
9.2.1	<i>Desarrollo de la aplicación para iOS</i>	111
9.2.2	<i>Pagos reales por la pasarela de pago de PayPal</i>	112
9.2.3	<i>Mejora del log de la aplicación</i>	114
9.2.4	<i>Realización de tests automáticos para el proyecto</i>	114
9.2.5	<i>Auditar las tarjetas de otras provincias</i>	117
CAPÍTULO 10.	PLANIFICACIÓN FINAL DEL PROYECTO	118
10.1	PLANIFICACIÓN DE LA AUDITORÍA.....	118
10.2	PLANIFICACIÓN DE LAS APLICACIONES ANDROID	118
CAPÍTULO 11.	PRESUPUESTO	124
11.1	PRESUPUESTO DE LA AUDITORÍA.....	124
11.1.1	<i>Gastos materiales</i>	124
11.1.2	<i>Gastos de recursos humanos</i>	125
11.1.3	<i>Gastos totales del desarrollo de la auditoría</i>	125
11.2	PRESUPUESTO DE LAS APLICACIONES ANDROID.....	126
11.2.1	<i>Archivos referenciados</i>	126
11.2.2	<i>Gastos materiales</i>	126
11.2.3	<i>Gastos de recursos humanos</i>	127
11.2.4	<i>Gastos totales del desarrollo de las aplicaciones Android</i>	128
11.3	PRESUPUESTO TOTAL	129
11.3.1	<i>Suma de los presupuestos de la auditoría y las aplicaciones Android</i>	129
11.3.2	<i>Gastos anuales a pagar por los servicios en la nube</i>	129
11.3.3	<i>Presupuesto total final</i>	131
CAPÍTULO 12.	REFERENCIAS BIBLIOGRÁFICAS	132
CAPÍTULO 13.	APÉNDICES	135
13.1	GLOSARIO Y DICCIONARIO DE DATOS	135

13.2 CONTENIDO ENTREGADO	136
13.2.1 Contenidos	136

Índice figuras

Figura 1 Esquema básico del funcionamiento de la aplicación.	6
Figura 2 Mapa de zonas de la CTA en Asturias.....	14
Figura 3 minSdkVersion del proyecto.	20
Figura 4 Índice de compatibilidad de las diferentes versiones de Android.	21
Figura 5 Archivo de planificación inicial de la auditoría (Parte 1).	25
Figura 6 Archivo de planificación inicial de la auditoría (Parte 2).	26
Figura 7 Archivo de planificación inicial de la auditoría (Parte 3).	26
Figura 8 Resumen de la planificación inicial de la auditoría.....	26
Figura 9 Presupuesto total estimado de la auditoría.	27
Figura 10 Planificación inicial del desarrollo de las aplicaciones (Parte 1).	27
Figura 11 Planificación inicial del desarrollo de las aplicaciones (Parte 2).	28
Figura 12 Resumen de la planificación de las aplicaciones para móviles.	28
Figura 13 Presupuesto total del desarrollo de las aplicaciones.	29
Figura 14 Suma de los presupuestos estimados de la auditoría y las aplicaciones.	29
Figura 15 Diagrama de casos de uso del sistema.	35
Figura 16 Todos los paquetes de PassengerApp mostrados en el explorador de paquetes de Android Studio.	40
Figura 17 Diagrama de paquetes de VehicleApp.	41
Figura 18 Diagrama de despliegue del sistema.	42
Figura 19 Diagrama de clases del módulo de autenticación.	45
Figura 20 Diagrama de clases del patrón singleton utilizado por las clases de la aplicación que desean interactuar con objetos comunes a toda la aplicación.	46
Figura 21 Diagrama de clases principal de la aplicación.....	47
Figura 22 Diagrama de secuencia para el caso de uso registrarse.....	48
Figura 23 Diagrama de secuencia para el caso de uso Iniciar sesión.	49
Figura 24 Diagrama de secuencia para el caso de uso cerrar sesión.	49
Figura 25 Diagrama de secuencia para los casos de uso establecer nombre, matrícula y número de zonas del vehículo.	50
Figura 26 Diagrama de secuencia para el caso de uso comprar bonos.	51
Figura 27 Diagrama de secuencia para el caso de uso Ver bonos comprados.	52
Figura 28 Diagrama de secuencia para el caso de uso Utilizar bonos.....	52
Figura 29 Diagrama de actividades del sistema.	53
Figura 30 Captura de pantalla del estado actual de la base de datos.....	54
Figura 31 Captura de pantalla de la colección “plates”.....	55
Figura 32 Captura de pantalla de los usuarios de la base de datos.	55
Figura 33 Formas de inicio de sesión permitidas por la base de datos.	56
Figura 34 Diagrama de la base de datos.	56
Figura 35 Información sobre Android Studio.	61
Figura 36 Botón de pruebas automatizadas.	69
Figura 37 Pantalla principal de las pruebas automatizadas.	70

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas.

Figura 38 Prueba de bluetooth fallida y exitosa.....	71
Figura 39 Test de inicio de sesión exitoso.....	72
Figura 40 Pruebas de la base de datos de Google Firebase exitosas.....	73
Figura 41 Cuenta destinada a testing en la consola de Google Firebase.....	74
Figura 42 Ejemplo de paleta de colores de la aplicación PassengerApp.....	80
Figura 43 La paleta de colores cumple los requisitos de accesibilidad.....	81
Figura 44 Pestaña gradle.....	83
Figura 45 Comando “gradle signInReport”.....	84
Figura 46 Contenido del hash “SHA1”.....	84
Figura 47 Botón “Agregar huella digital” en la configuración de la base de datos de Firebase.....	85
Figura 48 Pantallas comunes entre VehicleApp y PassengerApp que involucran la autenticación del usuario.....	87
Figura 49 Pantallas comunes de VehicleApp y PassengerApp que piden permisos al usuario para utilizar el Bluetooth y la localización del dispositivo.....	88
Figura 50 Pantalla principal de VehicleApp.....	89
Figura 51 Manual de usuario de VehicleApp.....	90
Figura 52 Pantalla principal de PassengeApp.....	91
Figura 53 Pantalla de “comprar bonos” de PassengerApp.....	92
Figura 54 Uso de PayPal para comprar 30 viajes de 1 zona.....	93
Figura 55 Mapa de zonas de la CTA en Asturias.....	94
Figura 56 Pantalla de “Mis bonos” de PassengerApp.....	95
Figura 57 Aplicación del pasajero buscando un vehículo.....	96
Figura 58 Aplicación del pasajero conectada a un autobús.....	97
Figura 59 Historial de viajes de la aplicación del pasajero.....	98
Figura 60 Bonos del usuario.....	99
Figura 61 Pasajero desconectándose manualmente del vehículo.....	100
Figura 62 Dispositivo del pasajero desconectado.....	101
Figura 63 Pantallas más relevantes de la aplicación del pasajero en inglés y con el tema oscuro.....	102
Figura 64 Pantallas más relevantes de la aplicación del pasajero en inglés y con el tema oscuro (Parte 2).....	102
Figura 65 Historial de la aplicación del pasajero en idioma inglés y tema oscuro.....	103
Figura 66 Paquetes principales PassengerApp.....	104
Figura 67 Paquete appTesting de PassengerApp.....	104
Figura 68 Paquete bluetooth de PassengerApp.....	105
Figura 69 Módulo protocol de PassengerApp.....	105
Figura 70 Paquete scan de PassengerApp.....	105
Figura 71 Raíz del proyecto PassengerApp.....	106
Figura 72 Diagrama de pantallas de PassengerApp.....	107
Figura 73 Diagrama de clases de VehicleApp.....	108
Figura 74 Diagrama de pantallas de VehicleApp.....	109
Figura 75 Centro de control de PayPal.....	113
Figura 76 Cuenta de desarrollador de PayPal, con dinero ficticio, el cual solo se permite gastar en aplicaciones que estén en modo de desarrollador (Sandbox).....	113
Figura 77 Principales tipos de test en Android.....	114
Figura 78 Resumen Unit tests.....	115
Figura 79 Resumen Instrumentation tests.....	116
Figura 80 Resumen UI tests.....	116
Figura 81 Archivo de planificación de las aplicaciones Android.....	119
Figura 82 Tareas comprendidas en la planificación de las aplicaciones Android (Parte 1).....	120
Figura 83 Tareas comprendidas en la planificación de las aplicaciones Android (Parte 2).....	121
Figura 84 División en roles de las tareas de la planificación.....	122

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas.

Figura 85 Resumen de la planificación de las aplicaciones Android.	123
Figura 86 Desglose del presupuesto de la auditoría.	126
Figura 87 Presupuesto final del desarrollo de las dos aplicaciones Android.	128
Figura 88 Resumen del presupuesto de las aplicaciones Android.	129
Figura 89 Cantidad de viajes llevados a cabo en Asturias durante los últimos años.	130
Figura 90 Carpetas principales del proyecto.	136
Figura 91 Archivos dentro de “Documentación App”.	136
Figura 92 Contenido de la carpeta “Archivos adjuntos”.	137
Figura 93 Contenido de la carpeta “Ayuda”....	137

Índice Tablas

Tabla 1 Requisitos del sistema globales (PassengerApp y VehicleApp).	32
Tabla 2 Requisitos del sistema para PassengerApp.	32
Tabla 3 Requisitos del sistema para VehicleApp.	33
Tabla 4 Caso de uso “Registrarse”.	35
Tabla 5 Caso de uso “Iniciar sesión”.	36
Tabla 6 Caso de uso “Cerrar sesión”.	36
Tabla 7 Caso de uso “Establecer nombre del vehículo”.	36
Tabla 8 Caso de uso “Establecer matrícula del vehículo”.	36
Tabla 9 Caso de uso “Establecer número de zonas del vehículo”.	37
Tabla 10 Caso de uso “Comprar bonos”.	37
Tabla 11 Caso de uso “Ver bonos comprados”.	37
Tabla 12 Caso de uso “Utilizar bonos”.	38
Tabla 13 Clase AuthActivity.	63
Tabla 14 Clase ChatServer.	64
Tabla 15 Clase BuyVouchersFragment.	65
Tabla 16 MyVouchersFragment.	65
Tabla 17 Clase DeviceScanFragment.	66
Tabla 18 Pruebas de Integración y del Sistema.	75

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La principal motivación del proyecto es auditar la seguridad de las tarjetas de transporte que utiliza la CTA (Consorcio de Transportes de Asturias), comprobando su seguridad ante clonados y, en caso de que esta seguridad sea débil, proveer una posible alternativa al sistema actual de tarjetas existente en el transporte público.

Como queda recogido en la noticia (El Comercio, 2018), existe un agujero de seguridad en esta tecnología, la cual hace factible la duplicación de tarjetas con una inversión mínima de dinero, y unos conocimientos informáticos medios. Tras los episodios producidos en 2018, debido a la vulnerabilidad conocida de estas tarjetas desde el año 2007, el objetivo del autor de este TFG es comprobar si esta vulnerabilidad ha sido subsanada y en caso contrario, desarrollar una alternativa factible, de bajo coste y con un mayor nivel de seguridad que el método empleado actualmente.

La alternativa propuesta se basaría en dos aplicaciones para Android que se comunicarían a través de la tecnología Bluetooth. Una irá instalada en el propio medio de transporte (autobús o tren), y otra irá instalada en el dispositivo móvil del pasajero.

Respecto al alcance del proyecto, se limita a proveer una prueba de concepto que permita demostrar que otras alternativas al uso de tarjetas NFC son viables, ya que el principal peso del proyecto recae en la auditoría y la investigación realizadas.

1.2 Resumen de Todos los Aspectos

Un resumen global de la aplicación se podría describir de la siguiente manera.

La alternativa propuesta se basaría en dos aplicaciones para Android que se comunicarían a través de la tecnología Bluetooth. Una irá instalada en el propio medio de transporte (autobús o tren), y otra irá instalada en el dispositivo móvil del pasajero.

Ambas aplicaciones están conectadas a una base de datos en la nube, proporcionada por Google Firebase.

La aplicación de los vehículos (autobuses o trenes) se llama VehicleApp. La misma permite al usuario (que será el conductor o responsable del bus o tren) seleccionar el número de zonas del bus o tren, su nombre (por ejemplo, “Autobús L5”, o “Tren C3 Gijón-Avilés” y su matrícula. Cuando la aplicación del vehículo está ejecutándose, activa el bluetooth y hace el dispositivo del medio de transporte visible.

El número de zonas de un vehículo se calcula teniendo en cuenta cuántos fragmentos del mapa Figura 2 se atraviesan en el transcurso de un viaje.



Figura 2 Mapa de zonas de la CTA en Asturias.

Tomando como ejemplo el mapa de zonas de la CTA, vemos que está fragmentado en parcelas más o menos similares a los concejos existentes aunque en algún caso uno de estos fragmentos se extiende sobre varios concejos. El número de zonas es el número máximo de fragmentos del mapa por las que puede pasar un determinado bus o tren en su itinerario. Por ejemplo, si el bus atraviesa 3 zonas, será necesario un bono de 3 o más zonas para poder utilizar ese bus.

La aplicación de los pasajeros se llama PassengerApp y es una aplicación que permite a los usuarios registrarse e identificarse, comprar bonos (a través de PayPal), ver los bonos que hay comprados, y consumir los bonos en viajes a realizar en autobuses o trenes. Para gastar un bono, la aplicación permite seleccionar en qué autobús o tren quiere el usuario gastar el bono. Los autobuses o trenes que el usuario tiene disponibles para gastar el bono son los que se

detectan por bluetooth, que tendrán el nombre y número de zonas que haya introducido el conductor en VehicleApp.

Una vez el usuario consume un bono se actualiza la base de datos para decrementarle un viaje al pasajero, y la pantalla del usuario se actualiza de manera que pueda demostrar al revisor que ha comprado el bono para realizar un viaje en esa línea como ticket justificativo. El pasajero tendrá a su disposición un historial de viajes que podrá utilizar con este mismo fin.

1.3 Lista de distribución

Todos los documentos entregados para este Trabajo de Fin de Grado están destinados a la siguiente audiencia:

- Alejandro León Pereira (Autor del TFG).
- Marco Antonio García Tamargo (Tutor del TFG).
- Cristian González García (Tutor del TFG).
- Jurado del TFG.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Actualmente, y como se ha demostrado en la auditoría realizada en este Trabajo de Fin de Grado (situada en Documentación Auditoría/Auditoría.pdf), y la cual es la parte principal del Trabajo de Fin de Grado, las tarjetas utilizadas por la mayoría de las compañías de transporte público, garajes y oficinas de todo el mundo (tarjetas denominadas técnicamente Mifare Classic) son vulnerables, pudiendo ser modificadas fácilmente por personas con el material y los conocimientos necesarios.

Estas vulnerabilidades podrían traer pérdidas a las compañías mencionadas anteriormente, por lo que el estudio de alternativas es necesario. En este proyecto se propone la alternativa de una aplicación para móviles que permita a los usuarios comprar sus bonos, administrarlos, y usarlos en autobuses o trenes compatibles con el sistema.

2.2 Objetivos del Proyecto

El proyecto tiene como objetivos los siguientes:

1. Comprender la estructura interna y la tecnología mediante la cual funcionan las tarjetas Mifare Classic mediante la auditoría realizada.
2. Demostrar la vulnerabilidad de las tarjetas Mifare Classic utilizadas por el Consorcio de Transportes de Asturias mediante la auditoría realizada.
3. Demostrar que se puede implementar una alternativa factible, económica y usable que aproveche las ventajas de la tecnología de radiofrecuencia Bluetooth, la cual está presente en la mayoría de los dispositivos móviles actuales.
4. Demostrar que el manejo de los bonos de transporte público, por parte tanto de los usuarios como de los trabajadores del sector, puede ser llevado a cabo de manera simple e intuitiva a través de una aplicación para móviles.

2.3 Estudio de la Situación Actual

Actualmente, al menos en España, no existen aplicaciones que permitan prescindir totalmente de tarjetas de transporte público. Lo que si existen son aplicaciones que permiten visualizar fácilmente las diferentes líneas de autobuses y trenes, pero no una aplicación que permita hacer uso de bonos directamente desde el móvil. Para llegar a esta conclusión hemos ido a los sitios web de las ciudades más grandes de España (Madrid, Valencia y Barcelona) y hemos revisado sus respectivos sitios web:

- Madrid: (Consorcio de Transportes de Madrid, 2022).
- Valencia: (S.A.U., 2022).
- Barcelona: (Grupo TMB, 2022).

Se han estudiado los métodos que se utilizan para gestionar los viajes de sus usuarios. En las tres ciudades se utilizan tarjetas, en ninguna se utilizan aplicaciones para móviles. En todo caso, las aplicaciones para móviles que se utilizan tienen como finalidad únicamente dar información acerca de los horarios de los autobuses y trenes, pero no permiten al usuario gestionar ni utilizar sus bonos del transporte público.

Utilizando las aplicaciones que ya hay para ver las diferentes líneas (como Google Maps) y la aplicación de este Trabajo de Fin de Grado, es posible prescindir totalmente del uso de tecnologías vulnerables como las tarjetas Mifare Classic (NXP Semiconductors, 2022), (nombre técnico por el cual se referencia a estas tarjetas de transporte público, garajes, etc.).

Actualmente los medios de transporte público, se utilizan o bien pagando con dinero en efectivo, o bien mediante el uso de unas tarjetas utilizadas por los pasajeros que, al pasarlas por un lector, hacen pitir al receptor (en el caso de un autobús) para que el conductor sepa que el pasajero tiene derecho a subir al autobús, o abren unas puertas para que, en el caso de los trenes, los pasajeros puedan acceder al apeadero. Las tarjetas se adquieren en estancos, o estaciones de autobús o tren. Actualmente la cantidad de viajes disponible en un determinado bono se encuentra almacenada en la información de las tarjetas. Esto supone una vulnerabilidad ya que el contenido de las tarjetas puede ser alterado, permitiendo a un usuario tener bonos infinitos sin pagar.

2.3.1 Evaluación de Alternativas

El proyecto a desarrollar consta de dos aplicaciones, una para ir instalada en un dispositivo Android que irá en el autobús o tren, y otra para ir instalada en los dispositivos Android de los pasajeros. Las dos aplicaciones se comunicarán entre sí para llevar a cabo el uso de bonos.

Dicho esto, la decisión más importante es decidir qué tecnología utilizar para la comunicación entre el dispositivo del vehículo, y el del pasajero. Se han barajado las alternativas explicadas en este apartado.

2.3.1.1 Conectividad

Respecto a la manera de conectar el dispositivo del vehículo con los dispositivos de los pasajeros, se ha estudiado tanto implementar dicha comunicación con NFC, como por Bluetooth. En este apartado se explicarán los pros y contras de cada tecnología.

2.3.1.1.1 NFC

NFC o Near Field Communication es la principal alternativa que se ha barajado para el proyecto. Es la tecnología que utiliza ahora mismo el sistema de transporte público, no obstante, tiene demasiados inconvenientes, comparada con el resto de las alternativas que se muestran a continuación.

Pros

La principal y única ventaja del NFC es que, al ser la tecnología actual, el sistema podría seguir teniendo los mismos lectores de tarjetas en los autobuses y trenes. No obstante, la razón de la existencia de la aplicación que se ha desarrollado es hacer frente a las vulnerabilidades de esta tecnología, por tanto, de seguir usando la misma tecnología de comunicación, se podrían seguir explotando vulnerabilidades en el sistema debido a la poca seguridad que ofrece el estándar.

Contras

Como ya se ha mencionado anteriormente, uno de los principales contras de NFC es la seguridad (Square Inc, 2022) de la organización mundial del NFC, en el que se habla de cómo se puede fácilmente interceptar las comunicaciones entre dos dispositivos NFC mediante un método llamado “escucha secreta”.

- En los dispositivos Android, “Android Beam” (Contribuidores Wikipedia, 2022) (estándar mediante el cual funciona NFC en el teléfono) se ha reemplazado en 2018 por “Nearby Share” (Joe Maring, 2022) otro estándar más novedoso, e incompatible con el anterior, haciendo que el desarrollo de la aplicación sea, no solo más costoso debido a que la tecnología Nearby Share es muy reciente y ofrece poca información y documentación en internet sino que, aparte, todos los dispositivos anteriores a 2018 serían incompatibles, dejando a una mayor parte de los pasajeros sin posibilidad de instalar la aplicación.

- Se podría desarrollar una aplicación que funcione en los dos estándares (tanto en Android beam como en Nearby Share), pero el coste de producción de la misma estaría fuera del alcance de los objetivos del proyecto, es decir, buscar una solución factible, usable por todos los pasajeros con un móvil estándar y que no supusiese un desembolso importante de terminales para las flotas.

- Problemas de incompatibilidad entre el lector del vehículo y el móvil del usuario, pues si el lector del vehículo es Android, no será compatible con dispositivos iOS (iPhone) con NFC, pues el NFC de Android y el de iPhone no son compatibles. Asimismo, si el lector del vehículo es iOS, no será compatible con dispositivos Android, como se muestra en el siguiente artículo (Jeremy O'Donoghue, 2021).

- Actualmente la tecnología NFC no suele venir implantada en los móviles del mercado, solamente modelos concretos traen NFC, por lo que la aplicación solo podría ser instalada por una minoría de los usuarios.

2.3.1.1.2 Bluetooth

Otra de las alternativas estudiadas es utilizar Bluetooth. El Bluetooth es una tecnología de transmisión de datos inalámbrica de medio alcance (hasta 6 o 7 metros) incorporada en el 100% de los teléfonos móviles que llevan saliendo en el mercado durante los últimos años. Para más información consultar la siguiente referencia: (Contribuidores Wikipedia, 2022).

Pros

Los pros de utilizar la tecnología bluetooth vienen dados por la compatibilidad, ya que el bluetooth es una tecnología madura, e implantada en todos los dispositivos móviles. A su vez, hay mucha documentación y ejemplos en internet para desarrollar aplicaciones basadas en Bluetooth, por lo que el coste del desarrollo se reduce enormemente.

- Todos los móviles tienen Bluetooth, por lo que, si le sumamos que la aplicación está desarrollada permitiendo la compatibilidad con dispositivos Android con una versión de API 23, es decir, los dispositivos con Android 6.0 (octubre de 2015) y superiores serán compatibles, tal y como muestra la siguiente referencia (Contribuidores Wikipedia, 2022). A continuación, se muestra en la Figura 3 la configuración del proyecto. El atributo “minSdkVersion” indica la versión de API mínima con la que funcionará nuestra aplicación:

```
defaultConfig {  
    applicationId "com.example.passengerapp"  
    minSdkVersion 23  
    targetSdkVersion 30  
    versionCode 1  
    versionName "1.0"
```

Figura 3 minSdkVersion del proyecto.

Observamos que obtenemos un índice de compatibilidad con 84.9% de todos los dispositivos Android del mundo, tal y como muestra la Figura 4:

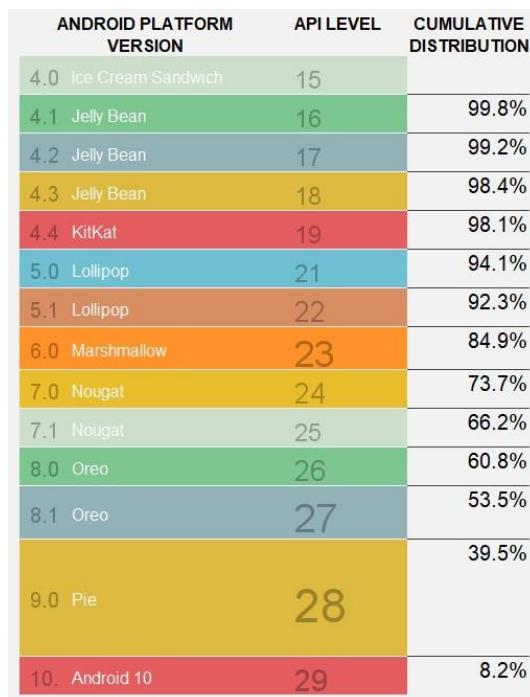


Figura 4 Índice de compatibilidad de las diferentes versiones de Android.

Si bien se podía haber utilizado un nivel de API 21, que cubriría el 94,1%, se ha optado por un nivel 23 ya que ha permitido implementar la autenticación por Google (Google, Integración de login con Google en aplicaciones, 2022) en la aplicación, facilitando al usuario la identificación mediante las cuentas de Google de su dispositivo, sin necesidad de introducir contraseñas.

- El bluetooth, a diferencia del NFC, es una tecnología totalmente compatible entre dispositivos Android e IOS. Por lo tanto, aunque los dispositivos de los vehículos sean Android, se podría desarrollar una aplicación para pasajeros con iPhone, y sería compatible con el sistema, ya que el iPhone del pasajero podría interaccionar con el Android del vehículo sin ningún problema.

Contras

Al cambiar la tecnología NFC por la tecnología Bluetooth, habría que sustituir los lectores NFC actuales de los vehículos (autobuses/trenes) por dispositivos Android, lo cual aumentaría notablemente el coste de implantación. No obstante, merece la pena debido a que la seguridad ayudará a compensar las pérdidas generadas por posibles mafias dedicadas a crear clones de tarjetas fraudulentas. Al final, el gasto en seguridad será una inversión que merecerá la pena.

Cabe decir que en el ámbito de la seguridad siempre se habla de dificultad a la hora de hackear, y no de si un sistema es hackable o no, ya que no existe ningún sistema que no se pueda hackear. Todos los sistemas se pueden hackear, aunque aún no haya sido descubierta la forma de hacerlo. Todos los sistemas del mundo son hackables, la diferencia es que unos son más difíciles de hackear que otros.

El Bluetooth, al igual que el NFC, se ve sometido constantemente a nuevas vulnerabilidades, tal y como muestra el siguiente artículo (Adrián Raya, 2022). No obstante, estas no son una amenaza para nuestro sistema, ya que la parte mediante la cual el sistema decide aprobar o denegar la petición de un usuario para subir a un vehículo, no se lleva a cabo a través de Bluetooth, sino de internet, mediante la base de datos de Google Firebase. Una vulnerabilidad en el Bluetooth ni siquiera podría llevar a un atacante a robar datos mediante un sistema de escucha, ya que todos estos datos se comunican a través de internet.

2.3.1.2 Base de datos

Para el desarrollo del proyecto es necesario alojar los datos de los usuarios en un servidor en la nube. Actualmente, el ámbito de los servicios en la nube está dominado por tres empresas principales :

- Amazon (Amazon Web Services).
- Google (Google Firebase).
- Microsoft (Microsoft Azure).

Para desarrollar el proyecto se ha escogido finalmente Google Firebase por las razones citadas a continuación (en resumen, su simplicidad y facilidad de uso, y que a pesar de no tener tantas configuraciones como Azure o Amazon Web Services, cumple los requisitos del proyecto ya que se trata de una aplicación sencilla).

2.3.1.2.1 Google Firebase

De todas las opciones de bases de datos disponibles en el mercado, se ha optado por Google Firebase. A continuación, se explican sus ventajas e inconvenientes respecto a otras bases de datos.

Pros

Las ventajas de utilizar Google Firebase son las siguientes:

- Abundancia de información y tutoriales en internet.
- Gratuito, y no es necesario proporcionar medios de pago para la magnitud de la aplicación desarrollada.
- Está respaldada por Google, y ofrece seguridad de serie de manera transparente para el programador, lo que disminuye el coste de desarrollo.
- Utilizar la base de datos de Google facilita la implementación de un sistema de autenticación mediante la cuenta de Google, a través del cual los usuarios podrán identificarse en la aplicación con la cuenta de Google del dispositivo, facilitando la interacción del usuario con el sistema.
- Sencilla de implementar.

Contras

Las desventajas son las siguientes:

- Para poner la aplicación en producción y abarcar un gran número de usuarios, habría que pagar ya que, a partir de cierto número de peticiones por día, la base de

datos es de pago. Cabe destacar que este contra se aplica también a las otras dos tecnologías barajadas: Microsoft Azure y Amazon Web Services.

2.3.1.2.2 Azure

Azure es la alternativa a Google Firebase que ofrece Microsoft para alojar servicios en la nube. Al igual que Google Firebase, Microsoft Azure permiten acceso gratuito para aplicaciones pequeñas, de manera que sea posible para programas pequeños y empresas emergentes empezar a usar sus servicios.

Pros

Las ventajas de Microsoft Azure se describen a continuación:

- El principal pro de la base de datos de Azure es que está respaldada por Microsoft, empresa grande y competente, por lo que los usuarios de Azure siempre dispondrán de documentación, tutoriales e información suficiente en internet para poder desarrollar las aplicaciones.
- La base de datos de Azure, al igual que Google Firebase, ofrece una capa de seguridad por defecto de manera transparente para el programador, lo cual disminuye los costes y tiempo de desarrollo.

Contras

Si bien la base de datos de Azure es una solución completamente factible para el desarrollo del proyecto, su manejo es mucho más difícil que el de Google Firebase. Si bien esto se debe a la cantidad de opciones que se brindan al usuario para dar libertad de poder hacer casi cualquier cosa, lo que a priori no es una desventaja, para el desarrollo del proyecto no es necesario una estructura compleja, por lo que la simplicidad de Google Firebase se compagina mejor con el proyecto.

2.3.1.2.3 Amazon Web Services

En comparación con la opción elegida finalmente, Google Firebase, Amazon Web Services tiene exactamente los mismos pros y contras que Microsoft Azure. Es decir, como pros, una gran empresa respaldando el servicio, y una seguridad implantada de serie, y como contras, una complejidad excesiva. O más bien, una orientación a proyectos de una mayor escala, disponiendo de un mayor número de opciones y configuraciones. Sin embargo, FireBase permite desplegar una base de datos sencilla en pocos pasos, lo cual para nuestro proyecto es una ventaja pues nuestra base de datos será básica.

Capítulo 3. Planificación del proyecto y presupuesto iniciales

Este punto corresponde tanto a un presupuesto, como a una planificación anteriores al desarrollo del proyecto, que servirá como una estimación de los tiempos que se tardarán en llevar a cabo el proyecto. En el caso de este Trabajo de Fin de Grado, se ha llevado a cabo una medición constante de los tiempos empleados en cada tarea llevada a cabo en el transcurso del proyecto. Esta medición correspondería a una planificación final y real del proyecto, y se describirá en detalle en el Capítulo 10. No obstante, a continuación, se encuentra la planificación inicial hecha anteriormente al desarrollo. Este apartado se divide en dos secciones, una para la planificación inicial de la auditoría, y otra para el desarrollo de las aplicaciones Android.

3.1 Planificación inicial de la auditoría

El archivo “Documentación App/Archivos adjuntos/Planificación y presupuesto iniciales/Planificación inicial auditoría.mpp” contiene el diagrama de Gantt de la planificación inicial de la auditoría, y se muestra en la Figura 5, Figura 6 y Figura 7:

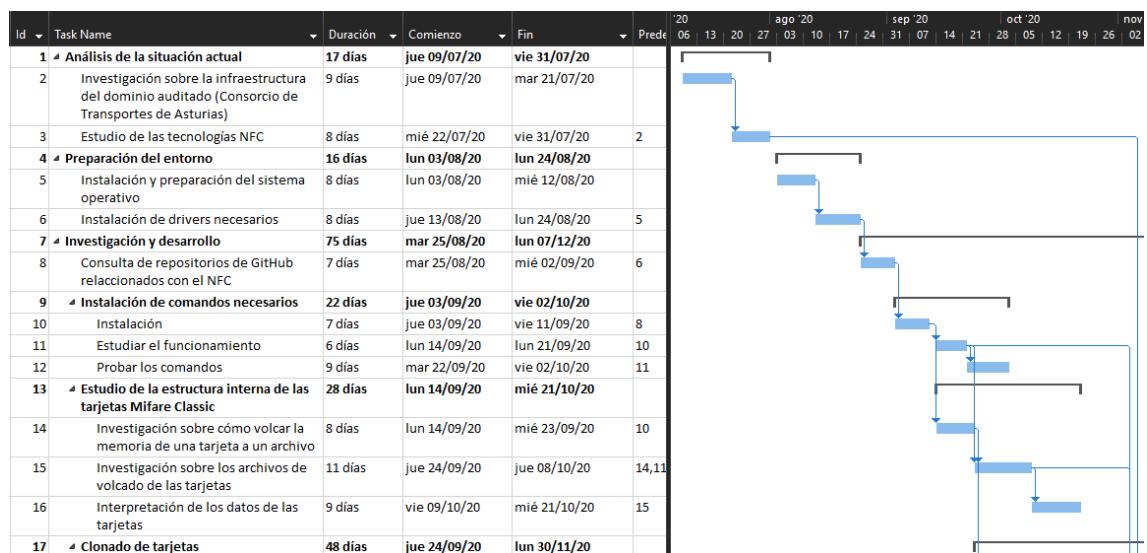


Figura 5 Archivo de planificación inicial de la auditoría (Parte 1).

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Planificación del proyecto y presupuesto iniciales

18	Investigación sobre cómo conseguir permisos de escritura en una tarjeta	11 días	jue 24/09/20	jue 08/10/20	14
19	Investigación sobre los sectores de la tarjeta que permiten lectura y escritura	18 días	vie 09/10/20	mar 03/11/20	18
20	Investigación sobre cómo sustituir el contenido de una tarjeta por el contenido de otra	11 días	mié 04/11/20	lun 30/11/20	19
21	▪ Generación de conclusiones	42 días	vie 09/10/20	lun 07/12/20	
22	Análisis de la seguridad y del cifrado de las tarjetas	11 días	vie 09/10/20	vie 23/10/20	18
23	Desarrollo de evidencias para demostrar las vulnerabilidades	10 días	mar 24/11/20	lun 07/12/20	22
24	Pruebas de lectura y escritura de diferentes tarjetas	12 días	mié 04/11/20	jue 19/11/20	11,15
25	Estudiar y valorar posibles alternativas al uso de tarjetas	11 días	vie 20/11/20	vie 04/12/20	24
26	Elaboración del presupuesto de la auditoría	6 días	lun 07/12/20	lun 14/12/20	24,25
27	Redacción del informe de auditoría	15 días	mar 15/12/20	lun 04/01/21	26
28	▪ Reuniones de seguimiento	42 días	lun 19/10/20	mar 15/12/20	
29	▪ Reunion de seguimiento 1	1 día	lun 19/10/20	lun 19/10/20	

Figura 6 Archivo de planificación inicial de la auditoría (Parte 2).

30	Elaborar acta de reunion	1 día	lun 19/10/20	lun 19/10/20	
31	▪ Reunion de seguimiento 2	1 día	vie 06/11/20	vie 06/11/20	
32	Elaborar acta de reunion	1 día	vie 06/11/20	vie 06/11/20	3
33	▪ Reunion de seguimiento 3	1 día	lun 16/11/20	lun 16/11/20	
34	Elaborar acta de reunion	1 día	lun 16/11/20	lun 16/11/20	
35	▪ Reunion de seguimiento 4	1 día	lun 23/11/20	lun 23/11/20	
36	Elaborar acta de reunion	1 día	lun 23/11/20	lun 23/11/20	
37	▪ Reunion de seguimiento 5	1 día	mar 15/12/20	mar 15/12/20	
38	Elaborar acta de reunion	1 día	mar 15/12/20	mar 15/12/20	

Figura 7 Archivo de planificación inicial de la auditoría (Parte 3).

El resumen de la planificación se encuentra en la Figura 8:

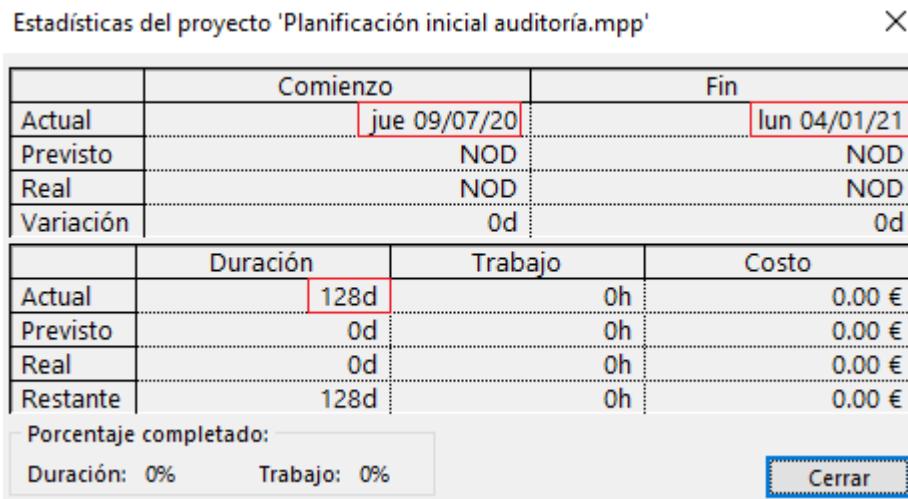


Figura 8 Resumen de la planificación inicial de la auditoría.

Como podemos observar en la Figura 8, se ha planificado realizar la auditoría entre las fechas resaltadas en rojo, jueves 9 de julio de 2020, y lunes 4 de enero de 2021. La duración total estimada sería de 128 días laborables.

Suponiendo una jornada laboral de 4 horas, en total serían 512 horas.

Tomando como precio de auditor 20€/hora, el presupuesto en relación con los gastos de recursos humanos sería 10.240,00€.

En total, incluyendo el IVA y el beneficio industrial, serían 13, tal y como muestra la Figura 9, procedente del archivo “Documentación App/Archivos adjuntos/Planificación y presupuesto iniciales/Presupuesto inicial.xlsx”.

RESUMEN DEL PRESUPUESTO: AUDITORÍA	
Coste Recursos Humanos	10.240,00 €
Beneficio Industrial (7%)	716,80 €
SUBTOTAL=	10.956,80 €
IVA 21%	2.300,93 €
TOTAL=	13.257,73 €

Figura 9 Presupuesto total estimado de la auditoría.

3.2 Planificación inicial de las aplicaciones Android

En el archivo “Documentación App/Archivos adjuntos/Planificación y presupuesto iniciales/Planificación inicial aplicaciones.mpp” se encuentra la planificación inicial de las aplicaciones para móviles. En la Figura 10 y Figura 11 se encuentra el diagrama de Gantt de dicho archivo, junto con sus tareas.

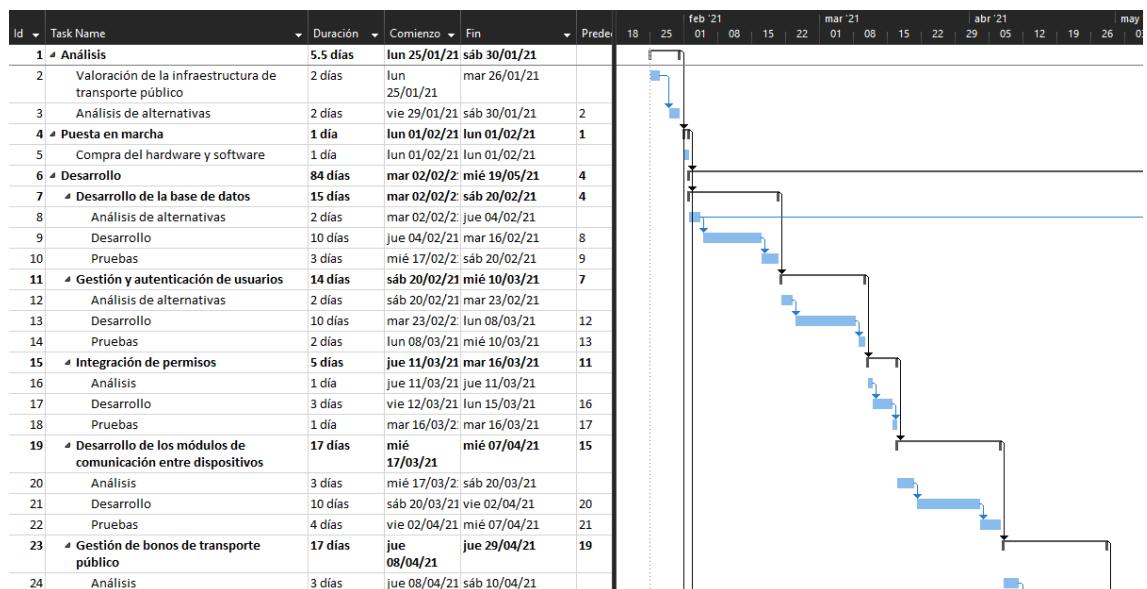


Figura 10 Planificación inicial del desarrollo de las aplicaciones (Parte 1).

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Planificación del proyecto y presupuesto iniciales

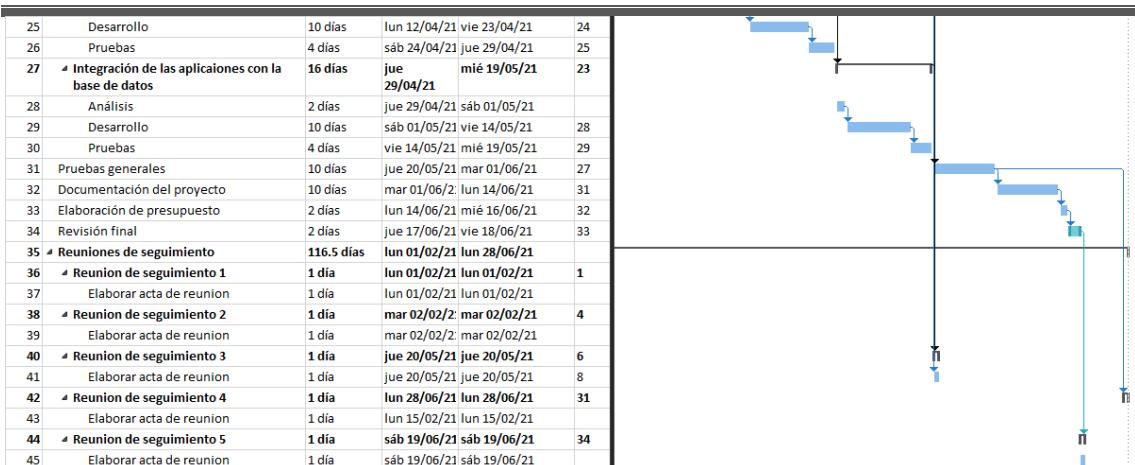


Figura 11 Planificación inicial del desarrollo de las aplicaciones (Parte 2).

En la Figura 12 se encuentra un resumen de la planificación.

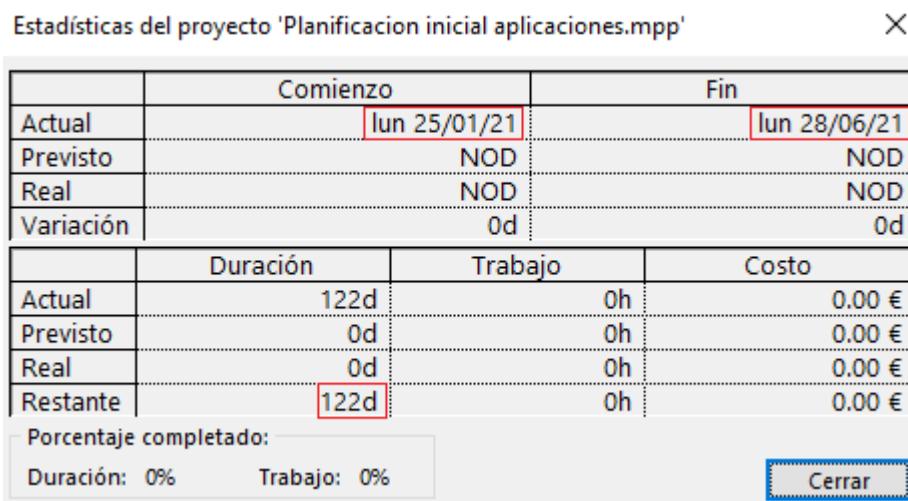


Figura 12 Resumen de la planificación de las aplicaciones para móviles.

Como podemos observar, la fecha de inicio es el lunes 25 de enero de 2021, y la fecha de fin prevista es el lunes 28 de junio de 2021. Si bien entre la fecha de inicio y la fecha de fin hay un total de 154 días, los días laborables son 122.

Teniendo en cuenta que se planea trabajar 4 horas por cada día laborable, las horas trabajadas en total sumarían: $122 \text{ días} * 4 \text{ horas/día} = 488 \text{ horas}$.

Tomando como precio por hora 20€/hora, el gasto total en recursos humanos serían: 488 horas * 20€/hora = 9.760,00€.

Como podemos observar en el fichero “Documentación App/Archivos adjuntos/Planificación y presupuesto iniciales/Presupuesto inicial.xlsx”, así como en la , el total del presupuesto de las aplicaciones sumaría un total de 12,636.27€, teniendo en cuenta el beneficio industrial y el IVA.

RESUMEN DEL PRESUPUESTO: APLICACIONES		
Coste Recursos Humanos	9.760,00 €	
Beneficio Industrial (7%)	683,20 €	
SUBTOTAL=	10.443,20 €	
IVA 21%	2.193,07 €	
TOTAL=	12.636,27 €	

Figura 13 Presupuesto total del desarrollo de las aplicaciones.

3.3 Presupuesto estimado total

La suma de los presupuestos estimados de la auditoría y las aplicaciones, es 25.894,00€, tal y como indica la Figura 14, procedente del archivo “Documentación App/Archivos adjuntos/Planificación y presupuesto iniciales/Presupuesto inicial.xlsx”.

RESUMEN DEL PRESUPUESTO: AUDITORÍA		
Coste Recursos Humanos	10.240,00 €	
Beneficio Industrial (7%)	716,80 €	
SUBTOTAL=	10.956,80 €	
IVA 21%	2.300,93 €	
TOTAL=	13.257,73 €	

RESUMEN DEL PRESUPUESTO: APLICACIONES		
Coste Recursos Humanos	9.760,00 €	
Beneficio Industrial (7%)	683,20 €	
SUBTOTAL=	10.443,20 €	
IVA 21%	2.193,07 €	
TOTAL=	12.636,27 €	

TOTAL APLICACIONES + AUDITORÍA	25.894,00 €
--------------------------------	-------------

Figura 14 Suma de los presupuestos estimados de la auditoría y las aplicaciones.

Capítulo 4. Análisis

Este apartado contendrá toda la especificación de requisitos y toda la documentación del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

4.1 Definición del Sistema

En este apartado se dará una definición del sistema muy por encima explicando sus objetivos y limitaciones.

4.1.1 Determinación del Alcance del Sistema

La aplicación desarrollada no es más que una prueba de concepto para demostrar que es posible sustituir el actual sistema de tarjetas como soporte de abonos de transporte público por una aplicación móvil.

En ningún caso la aplicación desarrollada está completamente lista para un despliegue real en trenes y autobuses, pues eso implicaría una labor mucho más compleja que la que se ha realizado en este Trabajo de Fin de Grado.

La aplicación desarrollada, permite a un usuario registrado simular la compra de bonos de transporte público y administrarlos. Dicha adquisición se hace por medio de una pasarela de pago de PayPal. La pasarela de pago está en modo desarrollador, por lo que se pueden utilizar cuentas de desarrollador con dinero ficticio. En el caso de que la aplicación se pusiera en producción, habría que ponerla en modo real, lo cual, una vez implementada la pasarela, no supone una gran complicación.

La aplicación permite iniciar sesión con correo electrónico o contraseña, así como cuenta de Google, y la contraseña estará cifrada para que posibles atacantes no puedan obtenerla, tal y como se haría en un entorno real.

El objetivo del proyecto es demostrar que es posible la comunicación por medio de Bluetooth entre un dispositivo emisor (el dispositivo Android del tren o autobús) y no uno, sino varios dispositivos receptores (los móviles de los pasajeros).

Si bien el principal objetivo del Trabajo de Fin de Grado es realizar una auditoría para demostrar la vulnerabilidad de las tarjetas NFC utilizadas por el transporte público en Asturias, se ofrece una alternativa a las mismas mediante la aplicación para Android desarrollada. La dificultad del desarrollo de las aplicaciones no es comunicar dos móviles entre sí por bluetooth, pues se conoce que eso es posible y de hecho hay muchos ejemplos en internet, sino comunicar un dispositivo bluetooth con varios al mismo tiempo, de manera segura y sin interferencias entre ellos.

4.2 Requisitos del Sistema

En este apartado, se explicarán los requisitos del sistema, así como la identificación de los actores del sistema y la especificación de los casos de uso.

4.2.1 Obtención de los Requisitos del Sistema

En este apartado se tratarán los requisitos del sistema funcionales, y no funcionales.

Requisitos del sistema funcionales

La Tabla 1 muestra una lista de los requisitos que el sistema debe cumplir para satisfacer las necesidades del cliente. Estos requisitos están orientados a ambas aplicaciones (PassengerApp y VehicleApp). Todos los requisitos tienen la misma prioridad pues son todos imprescindibles para el correcto funcionamiento del sistema.

Tabla 1 Requisitos del sistema globales (PassengerApp y VehicleApp).

Código	Descripción del Requisito
RAuth1	Se debe poder añadir un usuario al sistema una vez leídos y validados sus datos.
RAuth1.1	La autenticación podrá ser llevada a cabo por email o por cuenta de Google.
RAuth1.1.1	En el caso de la autenticación por email, el sistema verificará que el email es el del usuario mediante un correo de verificación.
RAuth1.1.2	En el caso de la autenticación por Google, el sistema verificará que el usuario existe consultando a los servicios de Google.
RAuth1.2	Los datos del usuario referidos en RAuth1 serán validados de acuerdo con unos estándares definidos por el administrador del sistema.
RAuth1.3	Si los datos referidos en RAuth1 cumplen la validación realizada en RAuth1.2, el sistema ingresará en la base de datos dichos datos.
RAuth1.3.1	Si la operación de escritura o actualización en base de datos no resultó exitosa se le mostrará al usuario un error y se le preguntará si desea intentar validarse de nuevo.
RAuth1.3.2	Si la operación de escritura o actualización en base de datos resultó exitosa, el usuario pasará a estar registrado en el sistema, y podrá iniciar sesión posteriormente (RAuth2).
RAuth2	Una vez añadido un usuario en RAuth1, dicho usuario debe poder iniciar sesión en el sistema.
RAuth2.1	Si el usuario referido en RAuth2 existe y la contraseña es correcta y corresponde a dicho usuario, el usuario pasará a estar autenticado.
RAuth3	El usuario podrá cerrar sesión.

Tabla 2 Requisitos del sistema para PassengerApp.

Código	Descripción del Requisito
RBonos1	Los usuarios autenticados en RAuth2.1 podrán comprar bonos de transporte público.
RBonos1.2	Los bonos referidos en RBonos1 se podrán comprar en packs de un número de bonos, modificable por el administrador, e inicialmente serán packs de 10, hasta un máximo de 50.

RBonos1.3	Una vez el usuario compre los bonos, dicha compra quedará registrada en la base de datos, para que posteriormente el usuario disponga de los bonos para consultarlos (RBonos2) y utilizarlos (RBonos3).
RBonos1.3.1	Si la base de datos no actualiza los datos exitosamente, se tomarán las acciones descritas en RAuth1.3.1.
Rbonos1.3.2	Si la base de datos actualiza los campos correctamente se le notificará al usuario de que pasará a tener disponibles los bonos.
RBonos2	Los usuarios autenticados en RAuth2.1 podrán consultar los bonos que han comprado en RBonos1.
RBonos3	Los usuarios autenticados en RAuth2.1 podrán utilizar los bonos disponibles en RBonos2 en los autobuses y trenes que estén lo suficientemente cercanos como para ser detectados por bluetooth.
RBonos3.1	El sistema mostrará al usuario una lista de autobuses o trenes en los que podrá gastar sus bonos.
RBonos3.2	Una vez utilizado un bono, el sistema mostrará al usuario la información conteniente en los siguientes subrequisitos.
RBonos3.2.1	Nombre del autobús o tren en el que se ha utilizado el bono.
RBonos3.2.2	Fecha y hora a la que ha sido utilizado dicho bono.
RBonos3.3	El bono tendrá validez y el usuario podrá viajar en el autobús o tren hasta el momento en el que se aleje lo suficiente como para que el autobús o tren pierdan la conexión con el dispositivo del usuario.
RBonos3.4	Una vez utilizado un bono la base de datos se actualizará restándole un bono.
RBonos3.4.1	En caso de error de la base de datos se llevarán a cabo las acciones descritas en RAuth1.3.1.
RBonos3.4.2	En caso de que la base de datos actualice los campos correctamente, se llevarán a cabo las acciones descritas en RBonos3.2.

Tabla 3 Requisitos del sistema para VehicleApp.

RVehiculo1	Los usuarios autenticados en RAuth2.1 podrán modificar los datos de los siguientes subrequisitos.
RVehiculo1.1	Nombre del vehículo.
RVehiculo1.2	Número de zonas del vehículo.
RVehiculo1.3	Matrícula del vehículo.
RVehiculo2	Una vez el usuario haya hecho alguno de los cambios descritos en RVehiculo1, la base de datos se actualizará en base a los mismos.
RVehiculo2.1	En caso de error de la base de datos se llevarán a cabo las acciones descritas en RAuth1.3.1.
RVehiculo2.2	En caso de que la base de datos actualice los campos correctamente, la aplicación se actualizará y el usuario podrá ver reflejados los mismos en la interfaz de usuario. Las aplicaciones de los pasajeros también verán los cambios reflejados a la hora de comunicarse con la aplicación del vehículo por bluetooth.

La Tabla 1, Tabla 2 y Tabla 3, se utilizan para mostrar todos los **requisitos funcionales** (lo que la aplicación debe hacer), mientras que los **requisitos no funcionales**, son los siguientes:

- **Requisitos de Usuario:** El usuario deberá estar registrado en algún proveedor de correo electrónico para poder utilizar la aplicación.

- **Requisitos Tecnológicos:** El sistema solo podrá funcionar en dispositivos Android en una versión Android 6 (Marshmallow) o superior. Asimismo, necesitará tener la ubicación del dispositivo activada y con permisos (que solicitará al usuario), así como el bluetooth.
- **Requisitos de Usabilidad:** La aplicación debe estar internacionalizada al menos en Inglés y Español, y debe poder ser utilizada tanto con un tema oscuro como con un tema claro, para que los usuarios con problemas de visión puedan utilizarla más fácilmente. Tanto el idioma como el tema serán escogidos por la aplicación automáticamente sin necesidad de acción del usuario, utilizando como referencia los ajustes actuales del sistema operativo del usuario.
- **Requisitos de Seguridad:** Los datos personales de los usuarios (email y contraseña) deberán estar cifrados para que no puedan ser descubiertos por atacantes, ya que, si el usuario utiliza la misma contraseña en nuestro sistema que en la cuenta bancaria, por ejemplo, una vulnerabilidad en el cifrado de las contraseñas de nuestro sistema podría permitir que un atacante obtenga acceso a la cuenta bancaria del usuario. Por ello tanto el email como la contraseña deben de estar asegurados.
- **Requisitos de Tiempo de Respuesta de la base de datos:** El sistema debe poder terminar las operaciones con la base de datos un 99% de las veces entre 0 y 2 segundos.
- **Requisitos de Tiempo de Respuesta de las conexiones Bluetooth:** El sistema deberá poder establecer conexiones bluetooth entre dispositivos el 99% de las veces entre 0 y 12 segundos. El tiempo medio deberá ser de 5 segundos, aunque pueda existir algún valor atípico, pero siempre inferior a 12 segundos.

4.2.2 Identificación de Actores del Sistema

Como ya se ha mencionado, el proyecto consta de dos aplicaciones, la aplicación que irá instalada en los dispositivos Android de los vehículos (autobuses y trenes), y la que irá instalada en los dispositivos Android de los pasajeros. No obstante, ambos comparten la misma base de datos. Dicho lo cual, los actores del sistema son los siguientes:

- Usuario pasajero no identificado.
- Usuario pasajero identificado.
- Usuario de vehículo (autobús o tren) no identificado.
- Usuario de vehículo (autobús o tren) identificado.
- Administradores de la base de datos..

4.2.3 Especificación de Casos de Uso

El diagrama de casos de uso del sistema se muestra en la Figura 15:

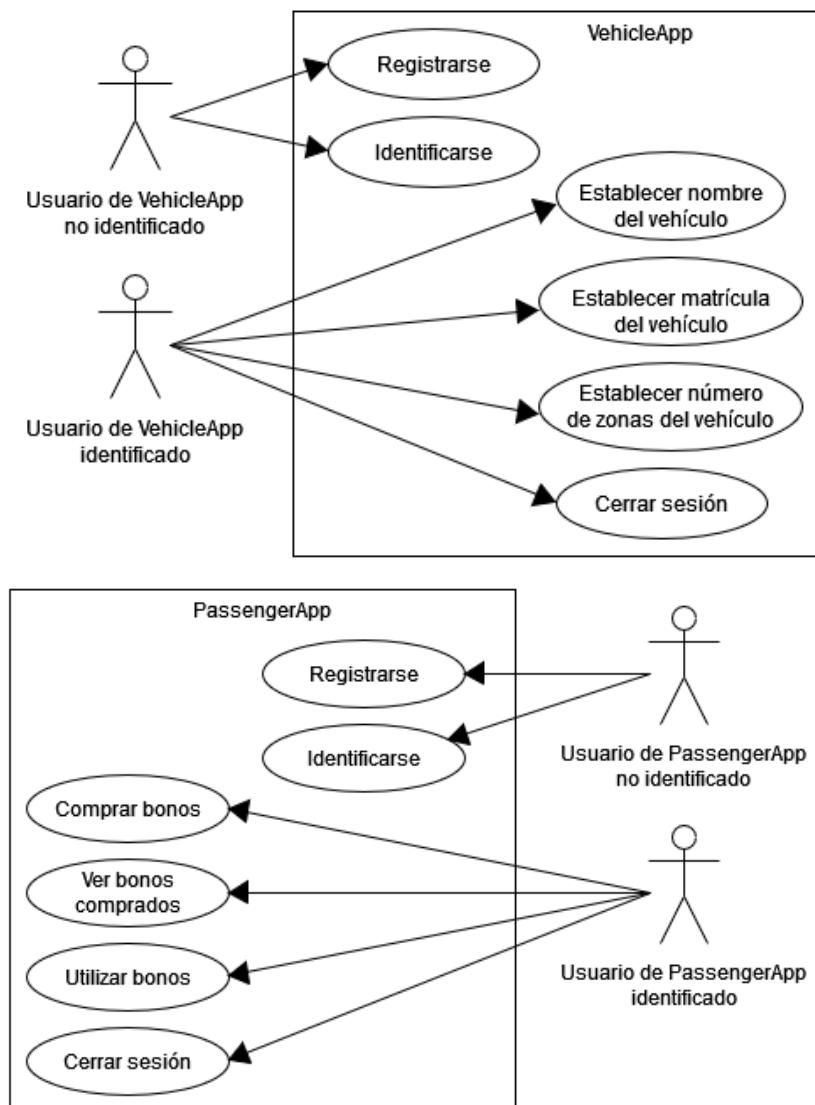


Figura 15 Diagrama de casos de uso del sistema.

A continuación, desde la Tabla 4 hasta la Tabla 12, se detalla una descripción de cada caso de uso.

Casos de uso globales (PassengerApp y VehicleApp)

Los casos de uso globales (tanto de PassengerApp como de VehicleApp) son los siguientes. Su código es CUGX (Caso de uso global).

Tabla 4 Caso de uso "Registrarse".

CUG1 - Registrarse
Precondición
El usuario deberá tener previamente una dirección de correo electrónico válida, o una cuenta de Google.
Descripción
Tanto un usuario de VehicleApp como de PassengerApp puede registrarse usando bien su email y una contraseña, o bien utilizando su cuenta de Google.
Postcondición

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Análisis

Una vez registrado se le creará un registro para él en la base de datos y podrá iniciar sesión posteriormente.

Tabla 5 Caso de uso “Iniciar sesión”.

CUG2 - Iniciar sesión
Precondición
El usuario debe haberse registrado previamente en la aplicación.
Descripción
Tanto un usuario de VehicleApp como de PassengerApp puede iniciar sesión en la aplicación. Si se ha registrado utilizando correo electrónico y contraseña deberá iniciar sesión utilizando ese método, y si se ha registrado utilizando su cuenta de Google deberá iniciar sesión con ella.
Postcondición
Si el usuario y contraseña son correctos, o la cuenta de Google existe y es válida, el usuario deberá ser redirigido a la pantalla principal de la aplicación.

Tabla 6 Caso de uso “Cerrar sesión”.

CUG3 - Cerrar sesión
Precondición
El usuario debe haber iniciado sesión previamente.
Descripción
Cualquier usuario tanto de PassengerApp como de VehicleApp podrá cerrar sesión.
Postcondición
Una vez cerrada la sesión, si quiere volver a utilizar la aplicación deberá volver a autenticarse ya sea con esa o con otra cuenta.

Casos de uso: VehicleApp

Los casos de uso referentes a la aplicación del vehículo son los siguientes.

Tabla 7 Caso de uso “Establecer nombre del vehículo”.

CUV1 - Establecer nombre del vehículo
Precondición
El usuario deberá haber iniciado sesión en la aplicación VehicleApp.
Descripción
Un usuario de VehicleApp registrado e identificado, podrá cambiar el nombre del vehículo que esté conduciendo.
Postcondición
El nombre influirá en cómo se “anunciará” a los pasajeros cuando detecten el vehículo por bluetooth.

Tabla 8 Caso de uso “Establecer matrícula del vehículo”.

CUV2 - Establecer matrícula del vehículo
Precondición
El usuario deberá haber iniciado sesión en la aplicación VehicleApp.
Descripción

Un usuario de VehicleApp registrado podrá cambiar la matrícula del vehículo que esté conduciendo. La matrícula no influirá en cómo se mostrará a los pasajeros cuando detecten el vehículo por bluetooth.

Postcondición

La matrícula se registrará en la base de datos por motivos de control.

Casos de uso: PassengerApp

Los casos de uso referentes a la aplicación del pasajero son los siguientes:

Tabla 9 Caso de uso “Establecer número de zonas del vehículo”.

CUV3 - Establecer número de zonas del vehículo
Precondición
El usuario deberá haber iniciado sesión en la aplicación VehicleApp.
Descripción
Un usuario de VehicleApp registrado podrá cambiar el número de zonas del vehículo que esté conduciendo.
Postcondición
El número de zonas influirá en cómo se mostrará a los pasajeros cuando detecten el vehículo por bluetooth, así como en el tipo de bono que se le restará al usuario cuando suba al autobús o tren. Por ejemplo, si el vehículo es de 2 zonas, se le restará al usuario uno de los bonos de 2 zonas que tenga comprados.

Tabla 10 Caso de uso “Comprar bonos”.

CUP1 - Comprar bonos
Precondición
El usuario deberá haber iniciado sesión en la aplicación PassengerApp.
Descripción
Un usuario de PassengerApp registrado podrá comprar bonos. Podrá comprarlos en packs de 10 (hasta llegar a 50) y de un número de zonas desde 1 zona hasta 7 zonas. Tanto los números de zonas como la cantidad de los packs y el límite podrá ser modificado por el administrador.
Postcondición
Los bonos comprados se verán reflejados en la base de datos, y el usuario los tendrá disponibles para su uso inmediatamente tras el momento de la compra.

Tabla 11 Caso de uso “Ver bonos comprados”.

CUP2 - Ver bonos comprados
Precondición
El usuario deberá haber iniciado sesión en la aplicación PassengerApp.
Descripción
Un usuario de PassengerApp registrado podrá ver qué bonos tiene comprados.
Postcondición
-

Tabla 12 Caso de uso “Utilizar bonos”.

CUP3 - Utilizar bonos
Precondición
El usuario deberá haber iniciado sesión en la aplicación PassengerApp.
Descripción
Un usuario de PassengerApp registrado podrá utilizar los bonos que tenga comprados en los vehículos que estén lo suficientemente cerca como para ser detectados por bluetooth. Solo podrá utilizar bonos en vehículos de su mismo número de zonas, por ejemplo, no podrá subir en un autobús de 2 zonas si solo tiene bonos de 1 zona.
Postcondición
Se le restará al usuario un bono de la zona correspondiente, en la pantalla principal se indicará en verde que se ha conectado al vehículo, y se añadirá una entrada en el historial de viajes para que el usuario pueda mostrársela al revisor en caso de que lo requiera.

Capítulo 5. Diseño del Sistema

En el apartado de diseño del sistema se explicará la arquitectura del mismo, el diseño de sus clases, los diagramas de estados, interacción y actividades, la base de datos utilizada en el sistema, y la interfaz de usuario.

5.1 Arquitectura del Sistema

En este apartado se comprenderán tanto la estructura de paquetes de las aplicaciones para Android desarrolladas como el esquema de despliegue de las mismas.

5.1.1 Diagramas de Paquetes

En este apartado se tratará la organización de las clases en paquetes dentro del proyecto. Se dividirá este punto en dos partes, una para PassengerApp y otra para VehicleApp.

5.1.1.1 Diagrama de paquetes de PassengerApp

En la Figura 16 se muestra una vista general de todos los paquetes de la aplicación PassengerApp:

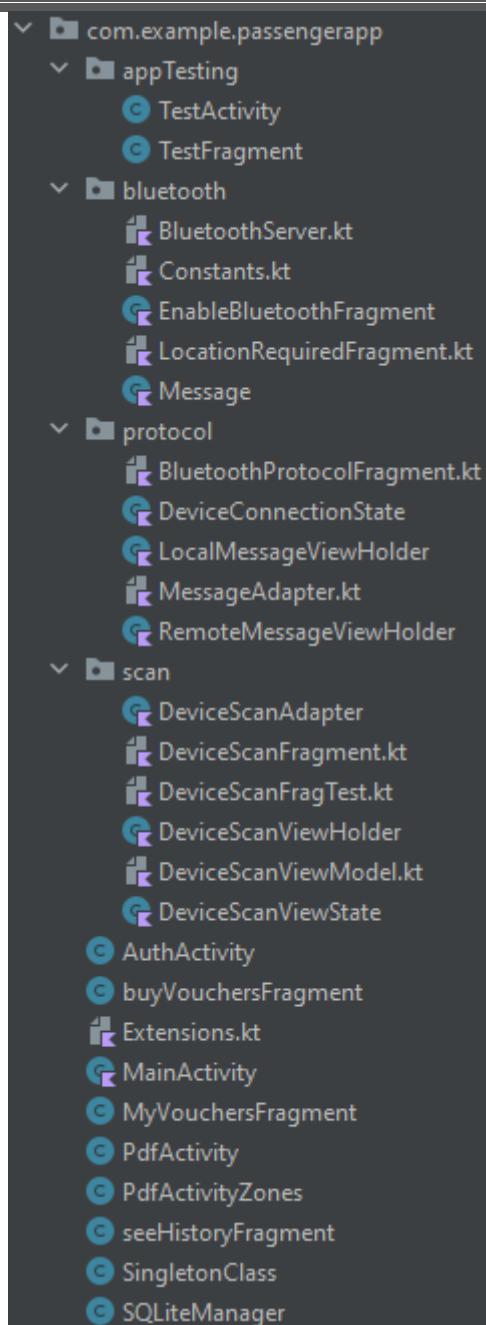


Figura 16 Todos los paquetes de PassengerApp mostrados en el explorador de paquetes de Android Studio.

Como podemos observar en la Figura 16, el proyecto se divide en cuatro principales paquetes, así como diferentes clases que están en la raíz del proyecto. Los cuatro paquetes presentes son los siguientes:

- appTesting: engloba las clases principales de los test de la aplicación.
- bluetooth: engloba todas las clases necesarias para establecer la comunicación Bluetooth entre dispositivos (ser descubribles y encontrar a otros dispositivos cercanos).
- protocol: engloba las clases necesarias para el intercambio de información entre dispositivos, una vez la conexión bluetooth ha sido realizada.

- scan: Engloba clases necesarias para escanear nuevos dispositivos mediante Bluetooth.

Las clases que están en el paquete principal (com.example.passengerapp) son clases que no encajan con ninguno de los paquetes mencionados anteriormente. Se tratará más en detalle la funcionalidad de dichas clases en el apartado 8.3.

5.1.1.2 Diagrama de paquetes de VehicleApp

El diagrama de paquetes de VehicleApp se muestra en la Figura 17:

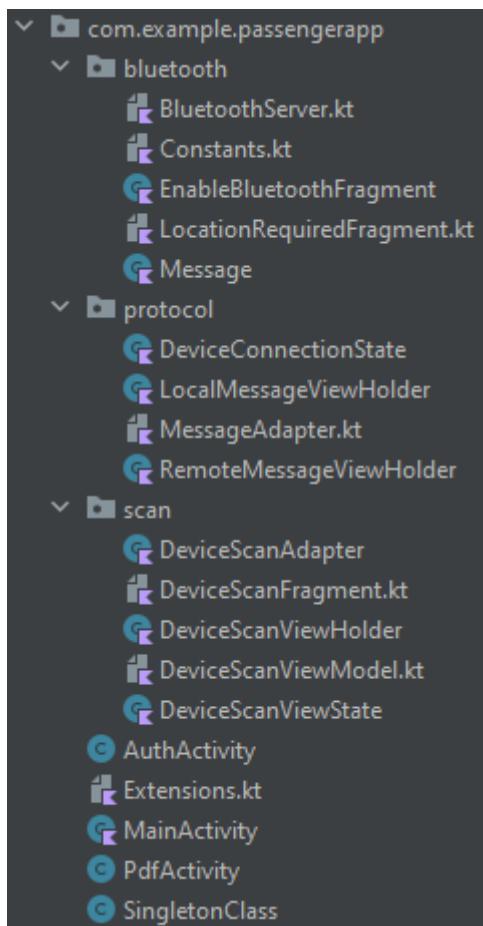


Figura 17 Diagrama de paquetes de VehicleApp.

Si bien la aplicación del vehículo tiene significativamente menos clases que la aplicación del pasajero, los paquetes “bluetooth”, “protocol” y “scan” mantienen las mismas funciones que las de la aplicación del pasajero, explicadas en el apartado 5.1.1.1. En el manual del programador situado en el apartado 8.3, se tratarán más en profundidad las funciones de dichas clases.

5.1.2 Diagramas de Despliegue

El sistema creado está compuesto por varios procesos *software* y máquinas que colaboran para llevar a cabo la tarea encomendada, la relación existente entre ellos es la mostrada en la Figura 18:

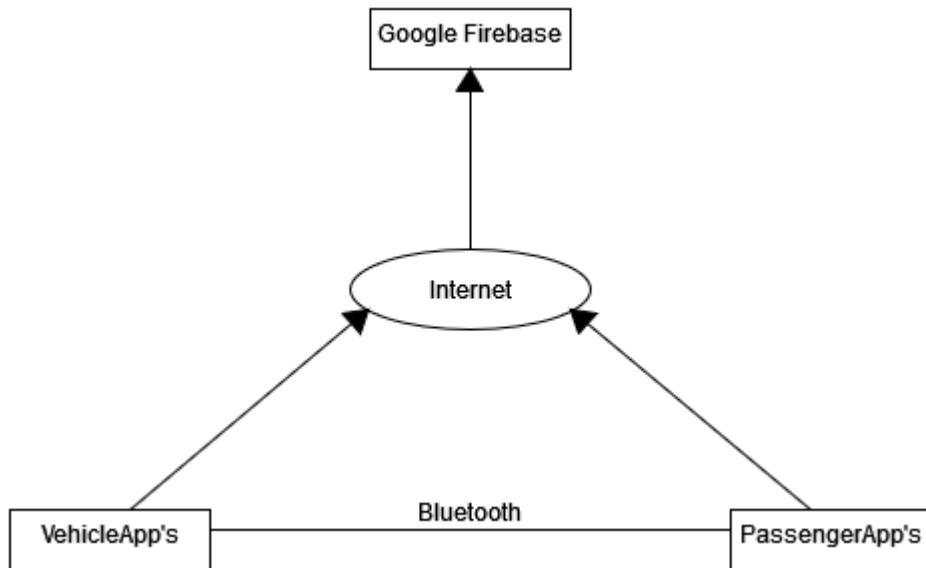


Figura 18 Diagrama de despliegue del sistema.

En la Figura 18 se ve como ambas aplicaciones (VehicleApp y PassengerApp) interactúan entre sí por bluetooth, y ambas interactúan con la base de datos de Google Firebase a través de internet. A continuación, se describe cada miembro de la Figura 18:

Google Firebase

Se trata de la base de datos donde está almacenada la información sobre el sistema. Dicha base de datos se describirá en detalle en el apartado 5.5.

VehicleApp's

Se trata de las aplicaciones instaladas en los dispositivos Android de vehículos.

PassengerApp's

Se trata de las aplicaciones instaladas en los dispositivos Android de pasajeros.

5.2 Diseño de Clases

Respecto a los diagramas de clases, se ha representado solamente un subconjunto de todas las clases de la aplicación, seleccionando las más relevantes con respecto al proyecto. A su vez, por simplicidad, solamente se mostrarán los diagramas para la aplicación de los pasajeros (PassengerApp), puesto que al ser las clases principales, y debido a la similitud existente entre las dos aplicaciones, el diagrama de clases resultante de la aplicación del vehículo (VehicleApp) sería exactamente igual, y por lo tanto resultaría redundante.

El diagrama de clases general es muy grande por lo que se ha dividido en dos diagramas, los diagramas de las clases más importantes de la aplicación.

El diagrama de las clases que se comunican con el módulo de autenticación es el mostrado en la Figura 19:

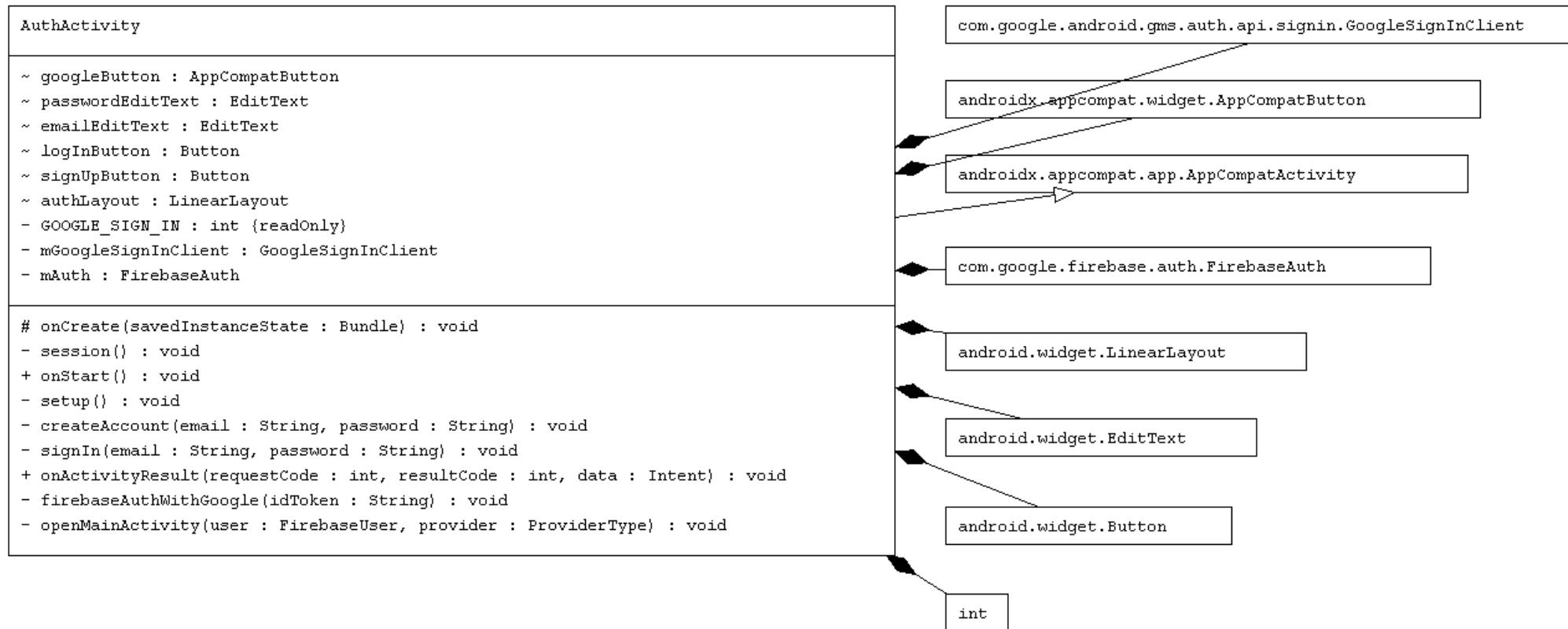


Figura 19 Diagrama de clases del módulo de autenticación.

El diagrama de clases mostrado anteriormente, es el que engloba todas las clases que utiliza el módulo de autenticación, que permite a los usuarios registrarse e iniciar sesión. Como podemos ver, la clase AuthActivity tiene los métodos createAccount() y signIn(), que son para crear una nueva cuenta y para iniciar sesión respectivamente.

El siguiente diagrama de clases es el “singleton”, el cual contiene los objetos comunes utilizados por todas las clases de la aplicación, y se muestra en la Figura 20:

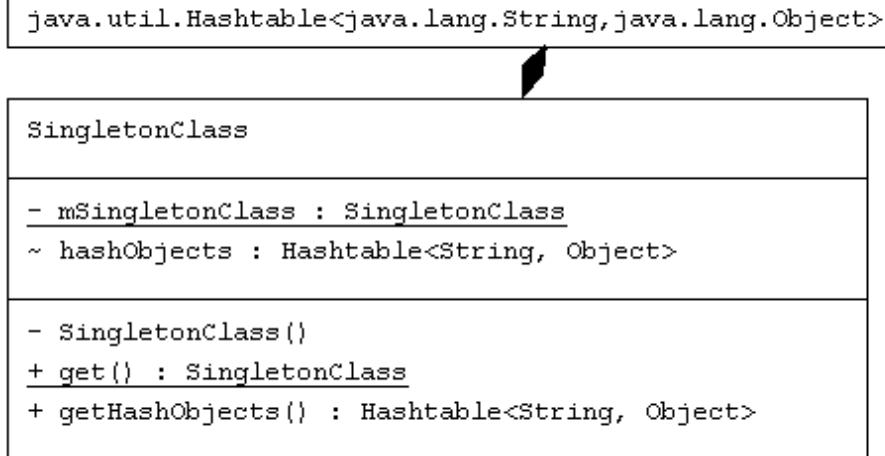


Figura 20 Diagrama de clases del patrón singleton utilizado por las clases de la aplicación que desean interactuar con objetos comunes a toda la aplicación.

Como podemos ver en la Figura 20, la clase “SingletonClass”, tiene un método “get()” que devuelve una instancia de la misma, y un método “getHashObjects()”, que es el método al que llaman las demás clases que quieren acceder a atributos comunes a toda la aplicación. Esos atributos comunes están almacenados en la “Hashtable” (o tabla hash) que devuelve dicho método.

El siguiente es el diagrama de clases conjunto de los tres módulos principales de la aplicación: el módulo de autenticación, la actividad principal de Android y el singleton, y se muestra en la Figura 21:

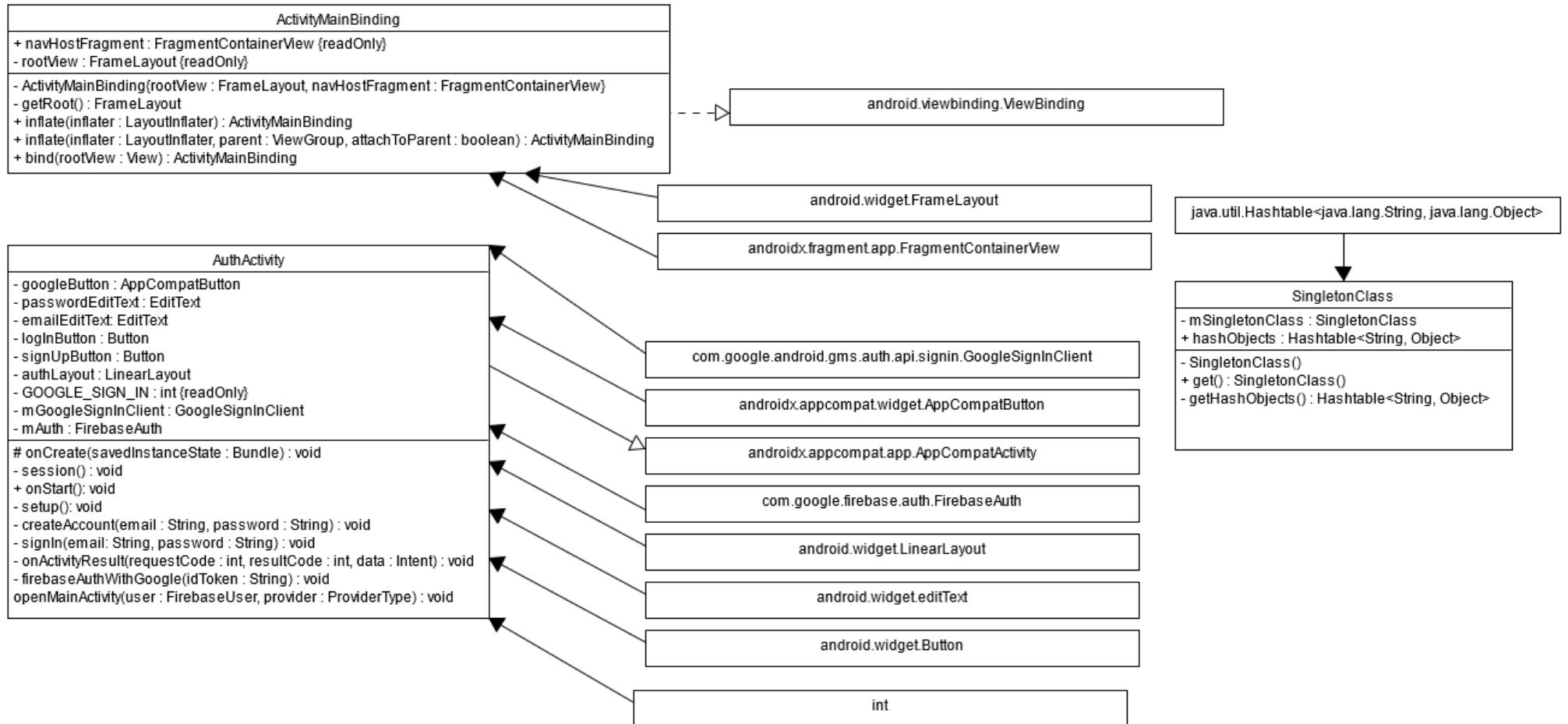


Figura 21 Diagrama de clases principal de la aplicación.

5.3 Diagramas de interacción y estados

En esta sección se mostrará un diagrama de interacción para cada caso de uso presentado en el punto [4.3.3 Especificación de Casos de Uso](#).

Los diagramas se han dividido en tres tipos: los diagramas comunes a las dos aplicaciones, los diagramas de PassengerApp y los diagramas de VehicleApp.

5.3.1 Diagramas comunes a las dos aplicaciones

En esta sección se engloban los diagramas que son exactamente iguales para ambas aplicaciones.

5.3.1.1 Caso de uso 1: Registrarse

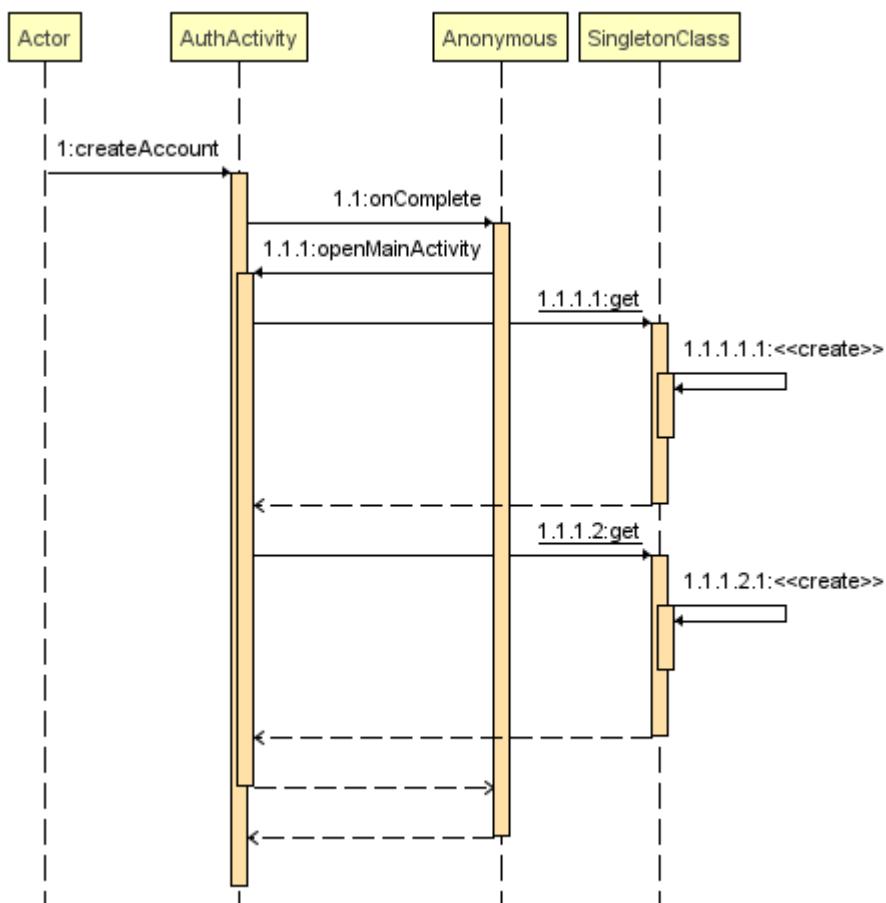


Figura 22 Diagrama de secuencia para el caso de uso registrarse.

Como podemos observar en la Figura 22, el método para crear una cuenta se comunica con AuthActivity y a su vez con SingletonClass, la cual tiene atributos comunes a toda la aplicación.

5.3.1.2 Caso de Uso 2: Iniciar sesión

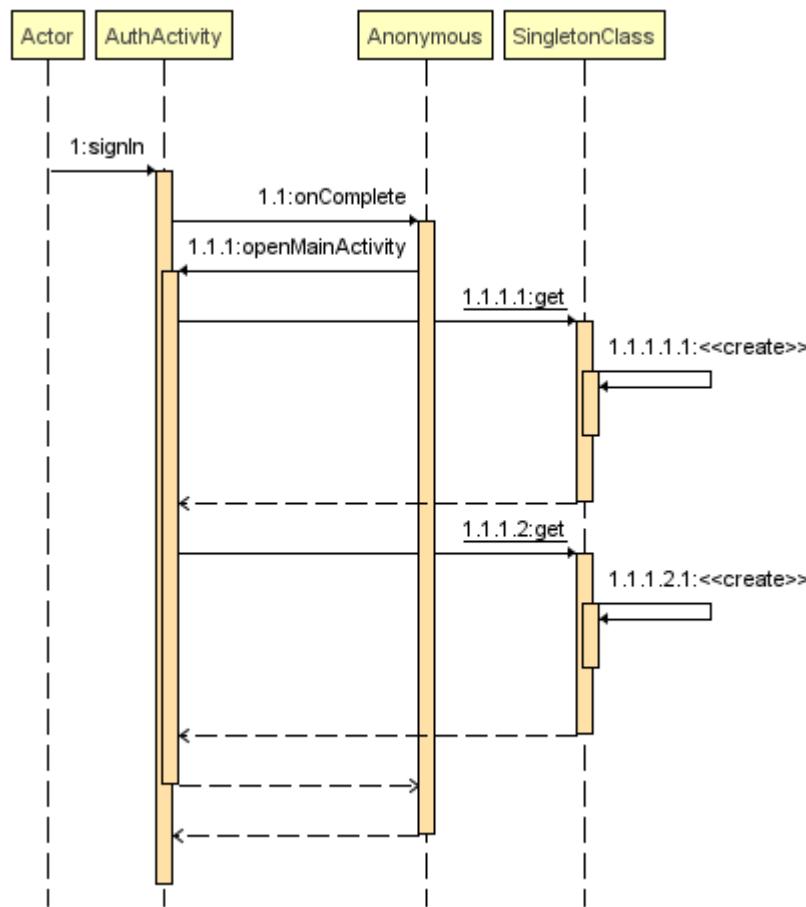


Figura 23 Diagrama de secuencia para el caso de uso Iniciar sesión.

El caso de iniciar sesión, mostrado en la Figura 23, es muy parecido al de crearse una cuenta. Al igual que en crear una cuenta, se comunica con la actividad de autenticación (AuthActivity) y luego con el singleton (SingletonClass).

5.3.1.3 Caso de uso 3: Cerrar sesión

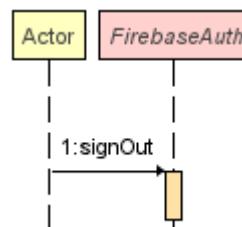


Figura 24 Diagrama de secuencia para el caso de uso cerrar sesión.

Como se muestra en la Figura 24, para cerrar sesión, simplemente se hace una llamada a FirebaseAuth mediante el método signOut(). Esto desvincula al usuario con la sesión existente en la nube de Google.

5.3.2 Diagramas de VehicleApp

En esta sección se mostrarán los diagramas de VehicleApp, la aplicación para autobuses y trenes.

5.3.2.1 Casos de uso: Establecer nombre, matrícula y número de zonas del vehículo

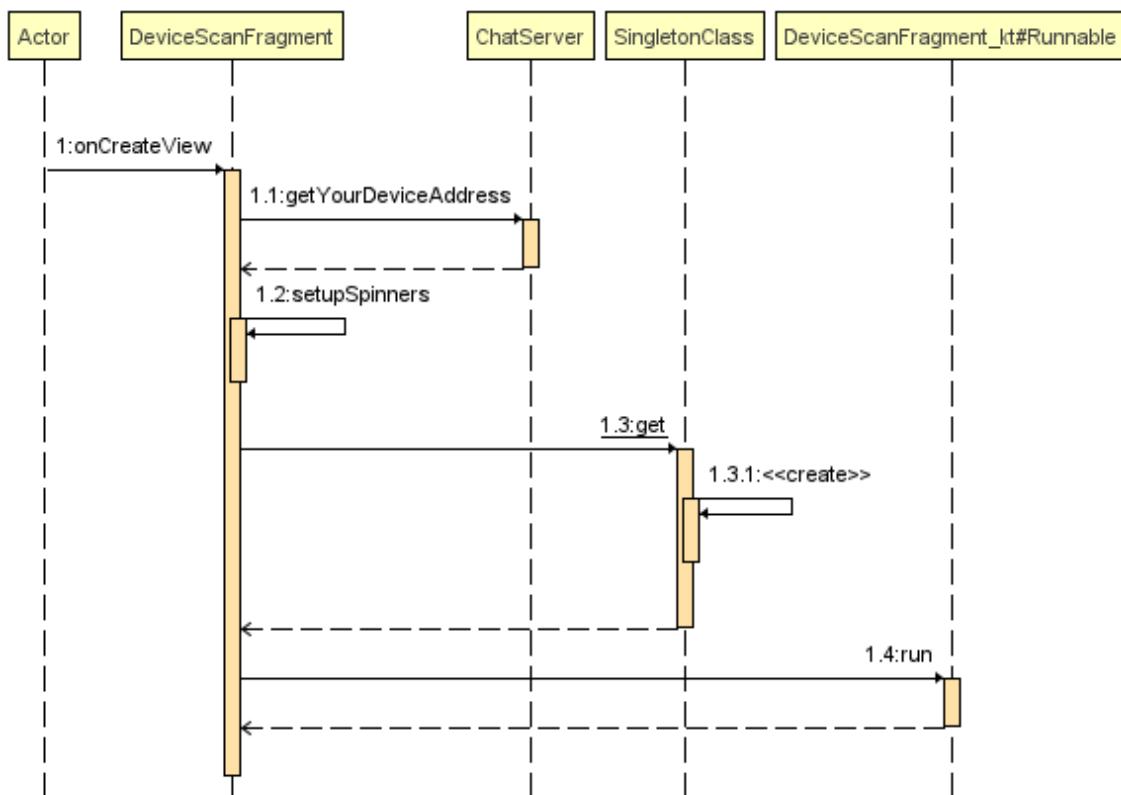


Figura 25 Diagrama de secuencia para los casos de uso establecer nombre, matrícula y número de zonas del vehículo.

Para los tres casos que se pueden dar en la aplicación del vehículo, se utiliza el mismo esquema, el mostrado en la Figura 25. Básicamente la clase SingletonClass tiene los atributos que se necesitan para que los tres casos puedan acceder a la base de datos a realizar las operaciones pertinentes.

5.3.3 Diagramas de PassengerApp

En esta sección se mostrarán los diagramas de PassengerApp.

5.3.3.1 Caso de uso 1 Comprar bonos

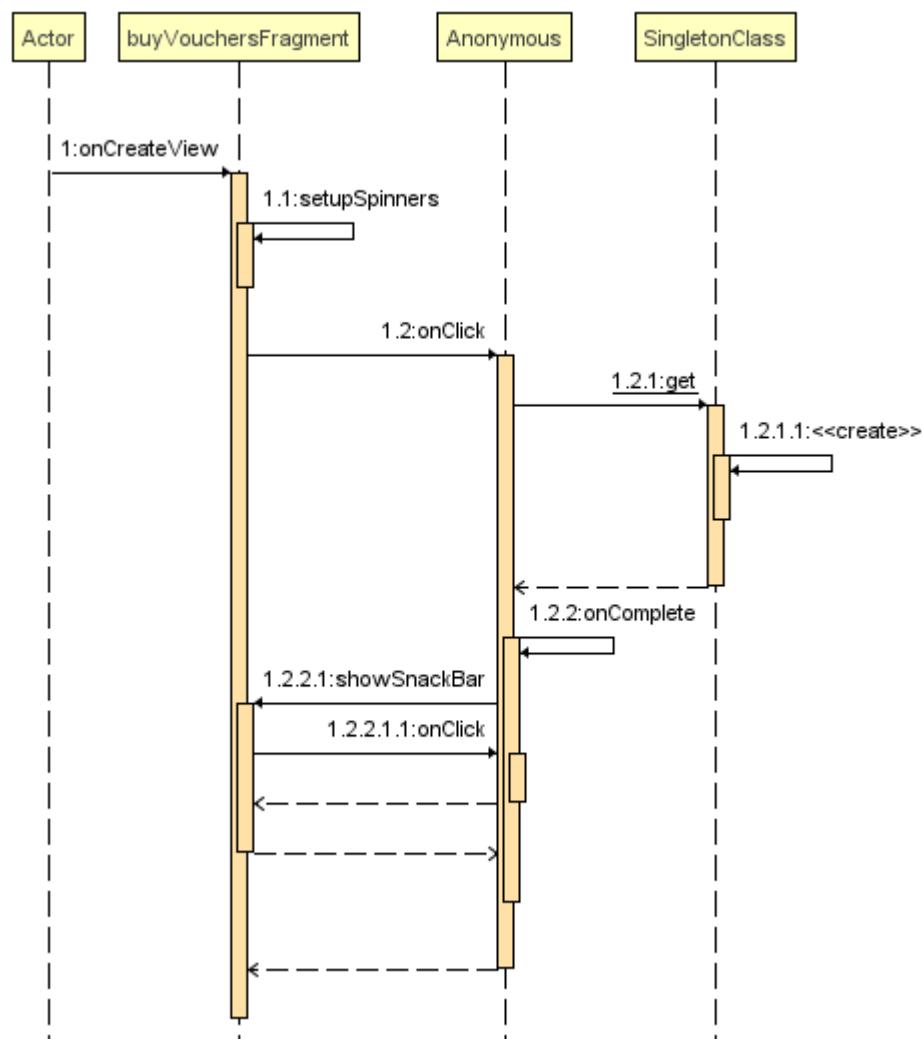


Figura 26 Diagrama de secuencia para el caso de uso comprar bonos.

En la Figura 26, la clase encargada de gestionar la compra de bonos es buyVouchersFragment. Se comunica a su vez con la clase SingletonClass para acceder a los atributos necesarios para hacer las operaciones pertinentes en la base de datos, y hace uso de la pasarela de pago de PayPal para proceder a la compra de bonos.

5.3.3.2 Caso de uso 2: Ver bonos comprados

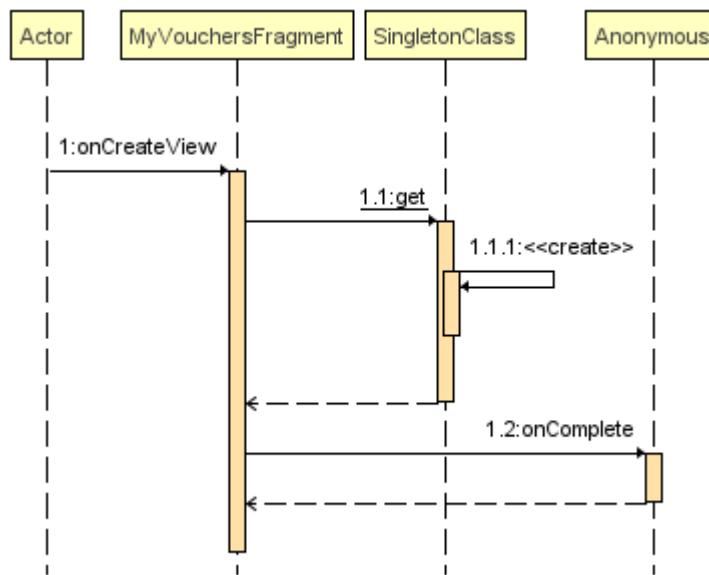


Figura 27 Diagrama de secuencia para el caso de uso Ver bonos comprados.

El caso dos, mostrado en la Figura 27, es bastante parecido al caso 1. La clase MyVouchersFragment, que es la encargada de mostrar al usuario sus bonos, se comunica con la clase SingletonClass para acceder a los atributos necesarios para operar con la base de datos y obtener los bonos del usuario.

5.3.3.3 Caso de uso 3: Utilizar bonos

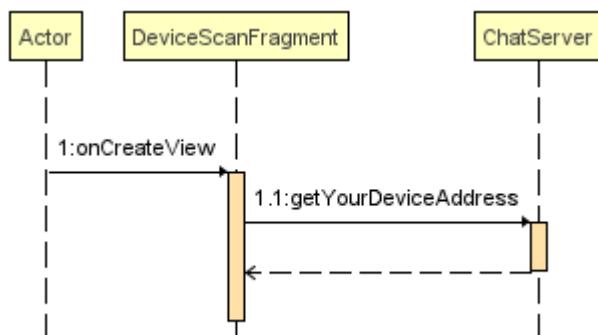


Figura 28 Diagrama de secuencia para el caso de uso Utilizar bonos.

En el caso 3, mostrado en la Figura 28, se ve que para utilizar los bonos es necesario hacer uso de la clase DeviceScanFragment, pues es la encargada de escanear en busca de vehículos. Después se comunica con la clase ChatServer para hacer el intercambio de información necesario para que la aplicación sepa de cuántas zonas es el vehículo (de 1, 2, 3, 4, 5, 6 o 7 zonas), para saber si el usuario tiene o no suficientes bonos para esa zona.

5.4 Diagramas de Actividades

A continuación, se muestra en la Figura 29 el diagrama de actividades del sistema en su conjunto:

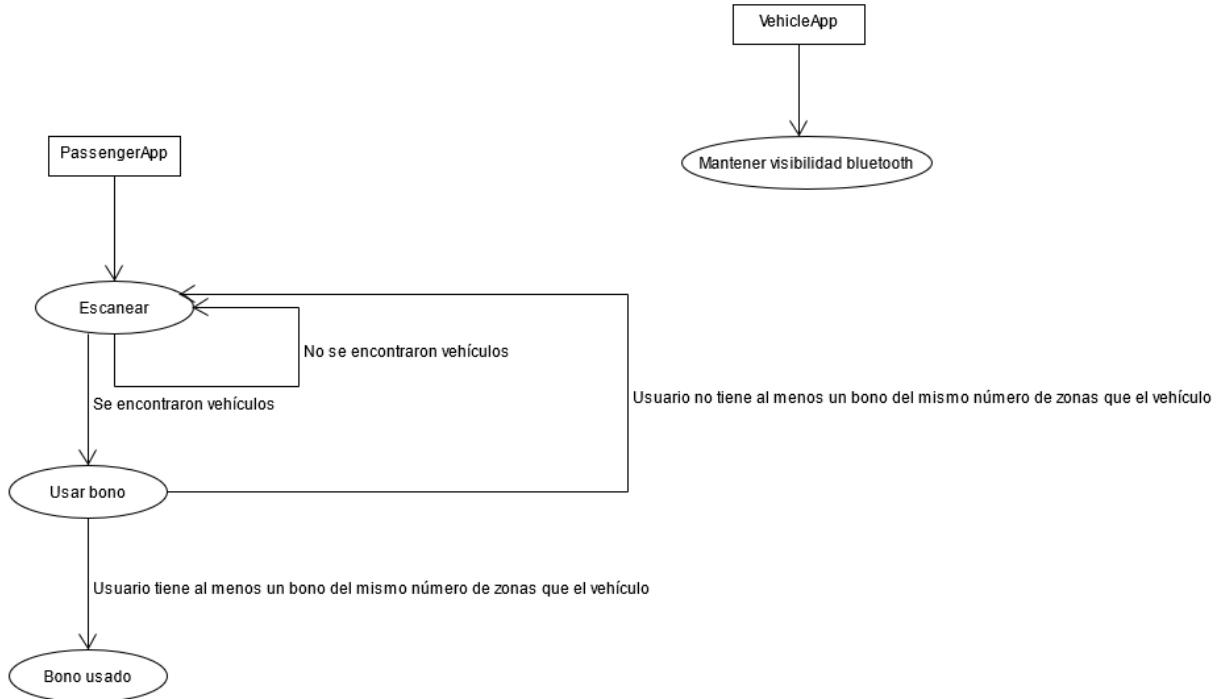


Figura 29 Diagrama de actividades del sistema.

Como podemos observar en la Figura 29, la aplicación VehicleApp se limita a mostrarse visible para dispositivos de pasajeros.

Es la aplicación de los pasajeros la que se encarga de escanear en busca de vehículos cercanos, y en caso de encontrarlos, dar la posibilidad al usuario de usar un bono, siempre que lo tenga disponible para el número de zonas del vehículo. Una vez usado, se le resta de sus bonos disponibles, y muestra por pantalla que el bono ha sido utilizado correctamente.

5.5 Diseño de la Base de Datos

Para el desarrollo del back-end de la aplicación se ha utilizado Google Firebase. La documentación de Firebase se puede encontrar en la siguiente referencia (Google, Documentación de Firestore, 2022).

Firebase permite sincronizar información en tiempo real con la aplicación Android, de manera que, al ser modificados los datos de la base de datos, se actualicen en tiempo real en la aplicación, y viceversa.

Para la identificación de los usuarios con la base de datos se ha utilizado la dependencia: “com.google.android.gms:play-services-auth:19.2.0”.

Respecto a la versión de Firebase, se ha utilizado la última versión a 24 de octubre de 2021.

5.5.1 Descripción del SGBD Usado

Firebase (La base de datos de Google) tiene dos principales tipos de bases de datos, Firestore y Realtime Database.

Se ha optado por Firestore por ser el tipo de base de datos más reciente de Firebase, habiendo dejado prácticamente obsoleta a Realtime Database, puesto que Firestore provee actualizaciones de datos en tiempo real, al igual que Realtime Database, pero con más ventajas, como por ejemplo una implementación mucho menos costosa.

5.5.2 Integración del SGBD en Nuestro Sistema

Nuestra base de datos consta de dos colecciones, “plates” y “users”, mostradas en la Figura 30.

users > alexleon8898@...		
tfgalp1	users	alexleon8898@gmail.com
+ Iniciar colección	+ Agregar documento	+ Iniciar colección
plates	alexleon8898@gmail.com >	+ Agregar campo
users >	alexleonperaaa3@gmail.com	tripsZone1: 40 tripsZone2: 20 tripsZone3: 37 tripsZone4: 8 tripsZone5: 8 tripsZone6: 0 tripsZone7: 0

Figura 30 Captura de pantalla del estado actual de la base de datos.

En la colección “users”, se almacenan los usuarios pasajeros. Cada usuario pasajero se identifica por su email, que será su clave. Cada usuario pasajero tiene almacenados los viajes que tiene comprados para cada tipo de bono. Por ejemplo, el usuario con email “alexleonperaaa3@gmail.com”, tiene 40 viajes para utilizar en autobuses o trenes de 1 zona, y 20 viajes para utilizar en buses o trenes de 2 zonas. Sin embargo, tiene 0 viajes de 6 zonas, por ejemplo.

plates			
tfgalp1	alexleonperaaa3@gmail.com		
+ Iniciar colección	+ Agregar documento	+ Iniciar colección	
plates >	conductordelbus@transporte1@gmail.com maxximumtrollers@gmail.com	+ Agregar campo	name: "Autobus L6 2zones" plate: "6782GDK"
users			

Figura 31 Captura de pantalla de la colección “plates”.

En la colección “plates”, mostrada en la Figura 31, tenemos por cada usuario conductor/maquinista (identificado por su email), el número de matrícula de su autobús o el número identificador del tren que está manejando: “plate”. Y el nombre de dicho autobús o tren.

En la base de datos se almacena cada usuario identificado por su email, como muestra la Figura 32:

Buscar por dirección de correo electrónico, número de teléfono o UID de usuario			
Identificador	Proveedores	Fecha de creación	↓ Fecha de acceso
georgeelchas@gmail.com	G	14 sep. 2021	15 sep. 2021
maxximumtrollers@gmail....	G	7 sep. 2021	23 sep. 2021
alexleon8898@gmail.com	G	6 sep. 2021	23 sep. 2021
alexleonperaaa3@gmail.co...	G	5 sep. 2021	23 oct. 2021

Figura 32 Captura de pantalla de los usuarios de la base de datos.

Para registrarse o iniciar sesión, se permite hacerlo a través de una cuenta de Google, o por correo electrónico y contraseña, de manera más convencional. En la Figura 33 se muestra que en la base de datos se admiten ambos métodos:

Proveedores de acceso	
Proveedor	Estado
✉ Correo electrónico/contraseña	Habilitado
Google	Habilitado

Figura 33 Formas de inicio de sesión permitidas por la base de datos.

El cifrado de la contraseña lo lleva a cabo automáticamente la base de datos de Google Firebase de manera transparente.

5.5.3 Diagrama E-R

La base de datos consta de dos tablas (users y plates) sin relación alguna entre ellas, mostradas en la Figura 34.

En la tabla users, se almacenan los viajes que tiene cada usuario pasajero.

En la tabla plates, se almacena la matrícula del vehículo del usuario conductor, así como el nombre de su vehículo, por ejemplo, Autobús L5.

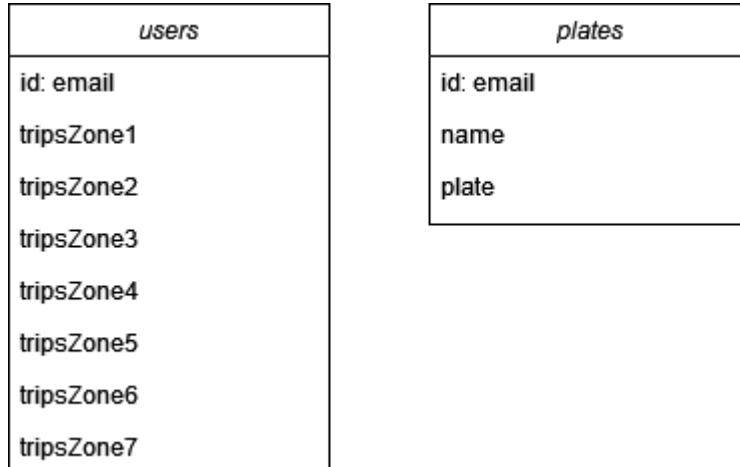


Figura 34 Diagrama de la base de datos.

5.6 Diseño de la Interfaz

Si bien no se ha realizado un diseño de la interfaz de usuario anterior al desarrollo de las aplicaciones, el resultado final junto con una explicación de cada pantalla se puede encontrar en el apartado 8.2.

Capítulo 6. Implementación del Sistema

6.1 Estándares y Normas Seguidos

Para el desarrollo de la aplicación se han seguido las denominadas “Java Code Conventions”, listadas en el siguiente artículo (Oracle, Java Code Conventions, 1997). Son unas convenciones que define Oracle, las cuales recomiendan al usuario cumplir varias características en el código, como por ejemplo usar “camel case” al definir métodos, variables y clases (un ejemplo de una variable en formato camel case es “listaProducción”, con la primera letra en minúscula, sin espacios, y las letras por las que comience una palabra en mayúsculas). Esto es para que el código escrito por distintos programadores, alrededor de todo el mundo, tenga el mismo aspecto.

Asimismo, para el desarrollo de la aplicación, se han tomado como referencia la siguiente guía de calidad básica de las aplicaciones, hecha por Google (Google, Calidad básica de las apps, 2021). En esta lista se definen unos criterios de calidad básicos que, como recomienda Google, deben de cumplir todas las aplicaciones.

6.2 Lenguajes de Programación

El desarrollo de las aplicaciones tanto del vehículo como del pasajero se ha hecho en Android, utilizando tanto el lenguaje de programación Kotlin (Jet Brains, 2022), como Java (Oracle, Sitio web de Java, 2022) para la lógica de negocio, y XML (Roche, 2000) para la interfaz de usuario.

6.3 Herramientas y Programas Usados para el Desarrollo

En este apartado se enumerarán y describirán las diferentes herramientas y programas utilizados durante el desarrollo del proyecto.

6.3.1 Android Studio

Para desarrollar las aplicaciones se ha utilizado Android Studio, cuyo sitio web oficial se encuentra en la siguiente referencia (Google, Sitio web principal de Android Studio, 2022). En la Figura 35 podemos encontrar información sobre la versión de Android Studio que hemos utilizado en el proyecto:



Figura 35 Información sobre Android Studio.

Android Studio corre a su vez en un equipo con Windows 10 Home (Microsoft, Sistema operativo Windows 10, 2022) instalado.

6.3.2 Git

Para llevar a cabo el control de versiones y el avance del proyecto se ha utilizado Git (Torvalds, 2022) (versión 2.30.0.2). El repositorio se encuentra subido a la plataforma github.com (León Pereira, 2022) y su visibilidad es pública, por lo que no es necesaria autorización para su descarga.

6.3.3 scrcpy

Para visualizar la pantalla de un móvil Android conectado a un ordenador en la pantalla del ordenador se ha utilizado scrcpy (Contribuidores scrcpy, 2022) (versión 1.20) (el nombre del programa viene del inglés Screen Copy).

6.4 Creación del Sistema

En este apartado se encuentran todos los aspectos planteados durante la implementación del proyecto.

6.4.1 Problemas Encontrados

Durante el desarrollo del proyecto han surgido varios problemas, listados a continuación.

6.4.1.1 Incompatibilidad con NFC

Respecto al NFC, no es un estándar común a todos los dispositivos como, si lo es por ejemplo el wifi o el bluetooth. Un receptor wifi no distingue si se está utilizando un móvil Android o iPhone, o que versión. Sin embargo, el NFC si lo hace, lo cual complica las cosas por las siguientes razones:

- En Android, Android Beam se remplaza en 2018 por Nearby Share, lo que hace que los dispositivos Android posteriores a 2018 sean incompatibles con los inferiores a esa fecha.
- En Apple, no se usa ni Android Beam ni Nearby Share, sino que se utiliza AirDrop, de Apple. Lo cual se traduce en incompatibilidades con dispositivos Android.
- No todos los móviles traen NFC incorporado por lo que si queremos reemplazar las tarjetas vulnerables por la aplicación móvil no podremos hacerlo pues no todo el mundo tendrá acceso a la aplicación.

Como se ha explicado en el apartado 2.3.1.1.2, la solución ha sido utilizar Bluetooth en lugar de NFC. Se eliminan los problemas tanto de compatibilidad como de disponibilidad por parte de los usuarios, pues todos los dispositivos Bluetooth son compatibles entre sí, y además todos los móviles tienen Bluetooth incorporado.

6.4.1.2 Comportamiento Bluetooth de la aplicación

El segundo problema encontrado al realizar el proyecto es que al ser el bluetooth una tecnología utilizada por todo el mundo, cuando los pasajeros escaneaban en busca de vehículos también les aparecían otros dispositivos bluetooth.

La solución ha sido modificar el adaptador bluetooth de la aplicación para que solo los dispositivos que estén ejecutando VehicleApp sean visibles para PassengerApp. Dicho de otra manera, PassengerApp solo detecta dispositivos que estén ejecutando VehicleApp, por lo que se obtiene el efecto esperado.

6.4.1.3 Gran número de pasajeros en los medios de transporte

Un problema que surgió a la hora del desarrollo ha sido que, en la tecnología bluetooth, tan solo se puede conectar un dispositivo con otro, y no puede haber varios dispositivos a la vez

conectados a otro. Sin embargo, en la vida real habrá varios pasajeros que se subirán a un determinado vehículo (autobús o tren).

La solución por la que se ha optado, es aprovechar todo el intercambio de información que permite la tecnología Bluetooth, en el momento anterior a una conexión. Es decir, utilizando solo la funcionalidad de ser descubrible. Cuando un dispositivo es descubrible, hay varios atributos que se pueden leer de ese dispositivo sin necesidad de conectarse a él, como son el nombre (una cadena de caracteres, suficiente para representar un nombre de vehículo y su número de zonas), y la dirección MAC.

Cuando varios pasajeros se suben a un vehículo, la aplicación del pasajero lee la información que la aplicación del vehículo está emitiendo (nombre del vehículo y número de zonas), sin necesidad de conectarse a él, por lo que varios pasajeros pueden leer la información. El resto de funcionalidad se lleva a cabo a través de la base de datos de Google Firebase, mediante la cual, una vez el pasajero selecciona un vehículo, se le restan los viajes de las zonas correspondientes, sin necesidad de conectarse a él por Bluetooth. En el momento en el que el dispositivo del pasajero deje de ser capaz de detectar al vehículo por Bluetooth, la aplicación mostrará que ya no se está abordo de ese vehículo, y el pasajero podrá ver en el historial la hora a la que se conectó a dicho vehículo.

6.4.2 Descripción Detallada de las Clases

En la siguiente sección se mostrarán las responsabilidades , atributos y métodos de las clases más importantes del proyecto, así como posibles observaciones. Se ha dividido esta sección en tres partes, una para las clases que son comunes a las dos aplicaciones (PassengerApp y VehicleApp), otra para PassengerApp, y otra para VehicleApp.

6.4.2.1 Clases comunes a las dos aplicaciones

En esta sección se recogerán las clases que comparten las dos aplicaciones (PassengerApp y VehicleApp).

6.4.2.1.1 Clase AuthActivity.java

Es la clase responsable de autenticar al usuario mediante su cuenta de correo electrónico y contraseña, así como de registrar a nuevos usuarios. La clase está escrita en el lenguaje de programación Java, y se muestra en la Tabla 13.

Tabla 13 Clase AuthActivity.

Nombre	Tipo	Descripción	Hereda de...
AuthActivity	Public	Autentica a usuarios	AppCompatActivity
<u>Responsabilidades</u>			
Número	Descripción		
1	Registrar usuarios		
2	Loguear usuarios		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos

private	void	createAccount	String email, String password
private	void	signIn	String email, String password
private	void	firebaseAuthWithGoogle	String idToken
Atributos			
Acceso	Modo	Tipo o Clase	Nombre
private		FirebaseAuth	mAuth
private		GoogleSignInClient	mGoogleSignInClient
private	final	int	GOOGLE_SIGN_IN
Observaciones			

6.4.2.1.2 Clase ChatServer.kt

Es la clase principal por la cual pasa toda la comunicación Bluetooth entre los dispositivos, estén utilizando PassengerApp o VehicleApp. Dicha clase, mostrada en la Tabla 14, administra los mensajes que se envían de un dispositivo a otro, así como toda información relevante acerca de los mismos, la cual se lista a continuación:

- Nombre del dispositivo (Por ejemplo: Autobús L5).
- En caso de ser VehicleApp, deberá comunicar el número de zonas del vehículo, pues será necesario para saber qué tipo de viajes le tenemos que restar al pasajero que se conecte a ese vehículo.

Tabla 14 Clase ChatServer.

Nombre	Tipo	Descripción	Hereda de...
ChatServer	Public	Comunica dispositivos a través de bluetooth	
Responsabilidades			
Número	Descripción		
1	Establecer la conexión entre dispositivos cercanos.		
2	Enviar e interceptar mensajes de información de dispositivos cercanos.		
3	Hacer el dispositivo visible, o invisible según sea necesario		
Métodos			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
private	void	startServer	
private	void	stopServer	
private	void	setCurrentChatConnection	BluetoothDevice device
private	void	connectToChatDevice	BluetoothDevice device
private	void	startAdvertisement	
private	void	stopAdvertising	
Atributos			
Acceso	Modo	Tipo o Clase	Nombre
private		FirebaseAuth	mAuth
private		GoogleSignInClient	mGoogleSignInClient
private	final	int	GOOGLE_SIGN_IN

Observaciones

6.4.2.2 Clases de PassengerApp

6.4.2.2.1 BuyVouchersFragment.java

Se encarga de establecer la conexión a la base de datos para incrementar los bonos del usuario, y se muestra en la Tabla 15.

Tabla 15 Clase BuyVouchersFragment.

Nombre	Tipo	Descripción	Hereda de...
BuyVouchersFragment.java	Public	Se comunica con la base de datos para incrementar los bonos del usuario	Fragment
<u>Responsabilidades</u>			
Número	Descripción		
1	Establecer la conexión con la base de datos		
2	Hacer consultas para modificar los valores de la base de datos correspondientes al usuario identificado		
3	Informar al usuario de cualquier posible error		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
private	void	onCreateView	LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
private		FirebaseFirestore	db
private		Spinner	spinnerZones
private		Spinner	spinnerTrips
private		Button	payButton
private		ConstraintLayout	parent
<u>Observaciones</u>			
En su método onCreateView se llevan a cabo todas las operaciones de lectura y escritura de la base de datos.			

6.4.2.2.2 MyVouchersFragment.java

Se encarga de establecer la conexión a la base de datos para obtener los bonos del usuario, y se muestra en la Tabla 16.

Tabla 16 MyVouchersFragment.

Nombre	Tipo	Descripción	Hereda de...

MyVouchersFragment.java	Public	Se comunica con la base de datos para incrementar los bonos del usuario	Fragment
Responsabilidades			
Número	Descripción		
1	Establecer la conexión con la base de datos		
2	Hacer consultas para obtener los valores de la base de datos correspondientes al usuario identificado		
3	Informar al usuario de cualquier posible error		
Métodos			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
private	void	onCreateView	LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState
Atributos			
Acceso	Modo	Tipo o Clase	Nombre
private		FirebaseFirestore	db
private		TextView	textView1Zone
private		TextView	textView2Zone
private		TextView	textView3Zone
private		TextView	textView4Zone
private		TextView	textView5Zone
private		TextView	textView6Zone
private		TextView	textView7Zone
Observaciones			
En su método onCreateView se llevan a cabo todas las operaciones de lectura de la base de datos para poder mostrarle al usuario qué bonos tiene disponibles.			

6.4.2.3 Clases de VehicleApp

En este apartado se engloban las clases más importantes de VehicleApp.

En este caso solo destaca una clase, la clase DeviceScanFragment.kt, mostrada en la Tabla 17. Es una clase escrita en lenguaje Kotlin, que es el lenguaje de programación por excelencia en Android junto con Java.

En ella reside la funcionalidad más importante de la aplicación VehicleApp, que es modificar el nombre, la matrícula y el número de zonas del vehículo a través de la base de datos de Google Firebase, así como ser descubrible vía Bluetooth.

Las otras funcionalidades de VehicleApp como hacerse visible mediante Bluetooth y autenticar al usuario ya han sido nombradas en el apartado 6.4.2.1.

Tabla 17 Clase DeviceScanFragment.

Nombre	Tipo	Descripción	Hereda de...
DeviceScanFragment.kt	Public	Escanea en busca de dispositivos y permite al usuario modificar el	Fragment

		nombre, la matrícula y el número de zonas de su vehículo (autobús o tren).	
<i>Responsabilidades</i>			
Número		Descripción	
1		Establecer la conexión con la base de datos para poder modificar el nombre, la matrícula o el número de zonas del vehículo.	
2		Permitir al usuario introducir los valores deseados para cada uno de los atributos mencionados en el punto anterior.	
3		Informar al usuario de cualquier posible error	
<i>Métodos</i>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
private	void	onCreateView	LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState
<i>Atributos</i>			
Acceso	Modo	Tipo o Clase	Nombre
private		FirebaseFirestore	db
private		String	name
private		String	zones
private		String	plate
private		BluetoothAdapter	bluetoothAdapter
private		var	firstTimeInSpinner
private		FirebaseUser	currentUser
private		String	currentUserEmail
<i>Observaciones</i>			
Al igual que en resto de fragmentos de la aplicación, la funcionalidad se concentra en el método onCreateView. En dicho método se llevan a cabo todas las operaciones de lectura y escritura de la base de datos para permitir al usuario modificar los atributos de su vehículo.			

Capítulo 7. Desarrollo de las Pruebas

7.1 Pruebas Unitarias

Las pruebas automatizadas del proyecto se encuentran integradas en la propia aplicación del pasajero (PassengerApp), en un botón que no está a la vista en la aplicación destinada al usuario. Para realizar las pruebas, se debe cambiar la visibilidad del botón con identificador “testApp” de “invisible” (tal y como muestra la Figura 36) a “visible”. De esa manera, se mostrará justo debajo de los botones del inicio de sesión.

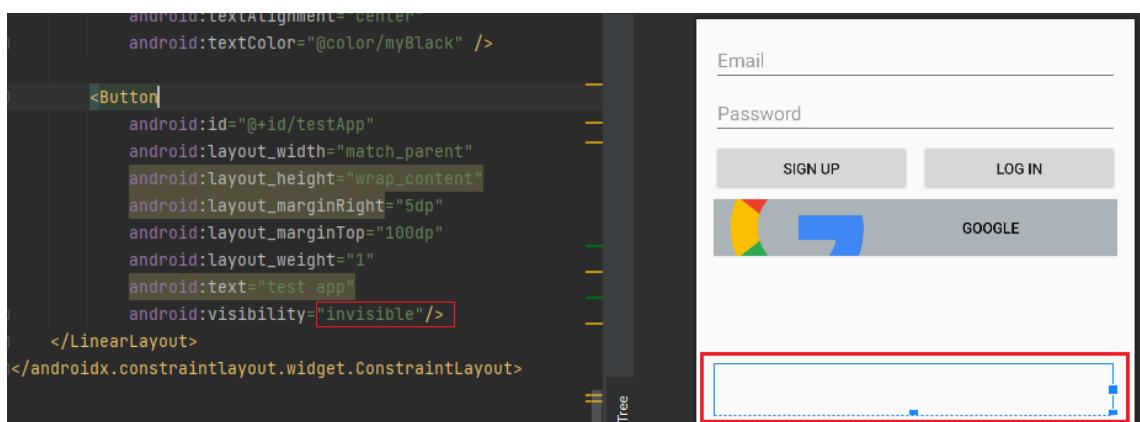


Figura 36 Botón de pruebas automatizadas.

Cabe destacar que, si bien se ha barajado la posibilidad de realizar las pruebas en unas clases de testing (JUnit), independientes de la aplicación, se ha decidido realizarlas dentro de la propia aplicación debido a que de esa manera se pueden probar tanto el Bluetooth, como Firebase, la base de datos de Google, la cual no permite hacer consultas desde una simple clase Java, ya que está configurada para admitir solamente dispositivos Android, instalados desde ordenadores autorizados con una determinada huella digital, tal y como se explicará en el manual de instalación, en apartado 8.1.

Una vez habilitado el botón de la Figura 36, desde la aplicación podremos acceder a las pruebas, tal y como se muestra en la Figura 37:

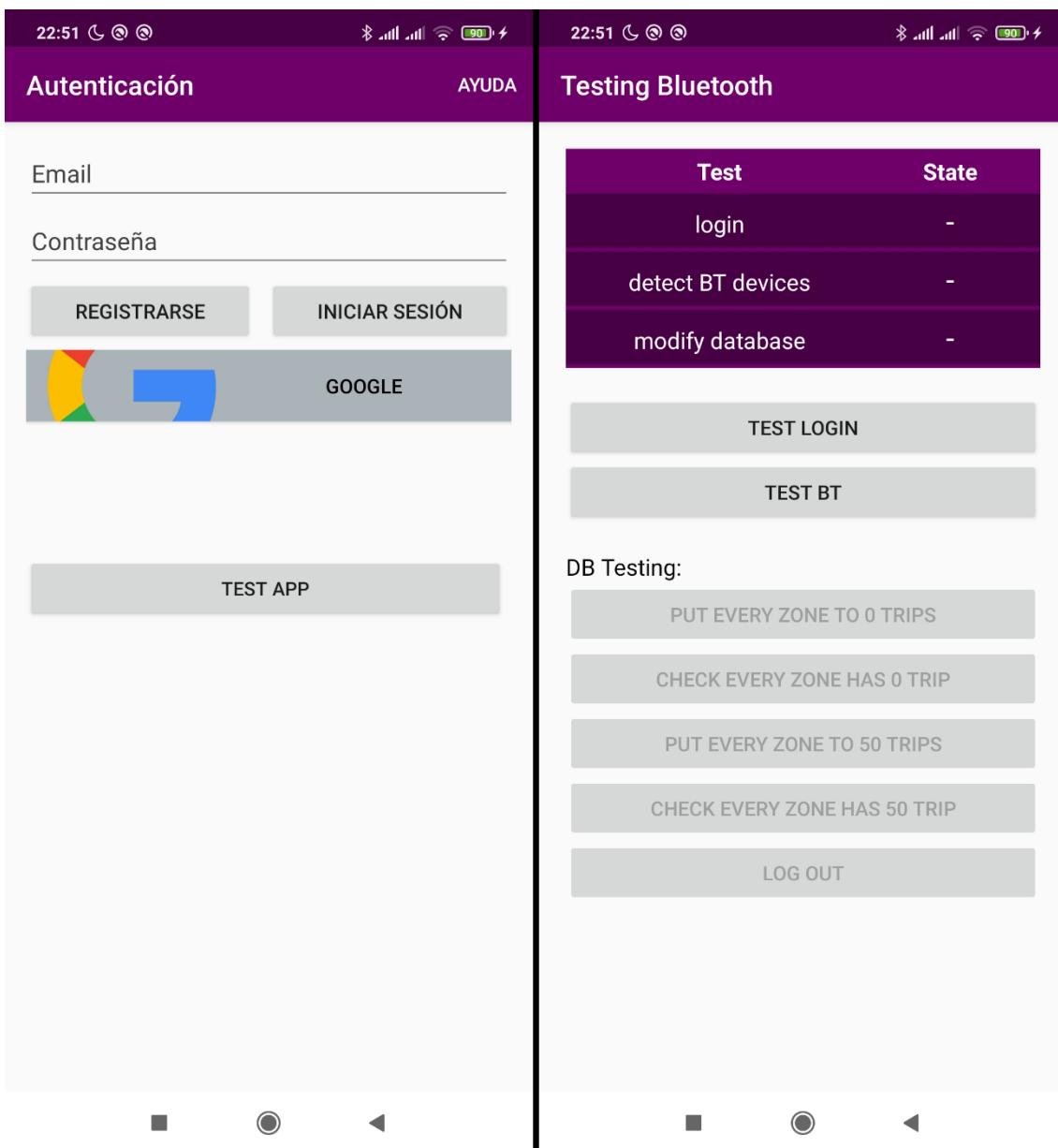


Figura 37 Pantalla principal de las pruebas automatizadas.

La Figura 37 muestra las funcionalidades disponibles. La tabla de color morado muestra el estado de cada test. Existen tres tipos de tests:

- Login: comprueba que el inicio de sesión con Google funciona correctamente.
- Detect BT devices: comprueba que la aplicación es capaz de detectar vehículos por bluetooth. Para que esta prueba termine correctamente es necesario disponer de un dispositivo con la aplicación VehicleApp ejecutándose.
- Modify database: comprueba que la aplicación es capaz de realizar operaciones sobre la base de datos de Google Firebase.

Al darle al botón “Test BT”, se probará la funcionalidad del Bluetooth. Si es capaz de encontrar un dispositivo, mostrará un “OK” verde en la tabla, y de lo contrario, mostrará un “KO” rojo, tal y como muestra la Figura 38:

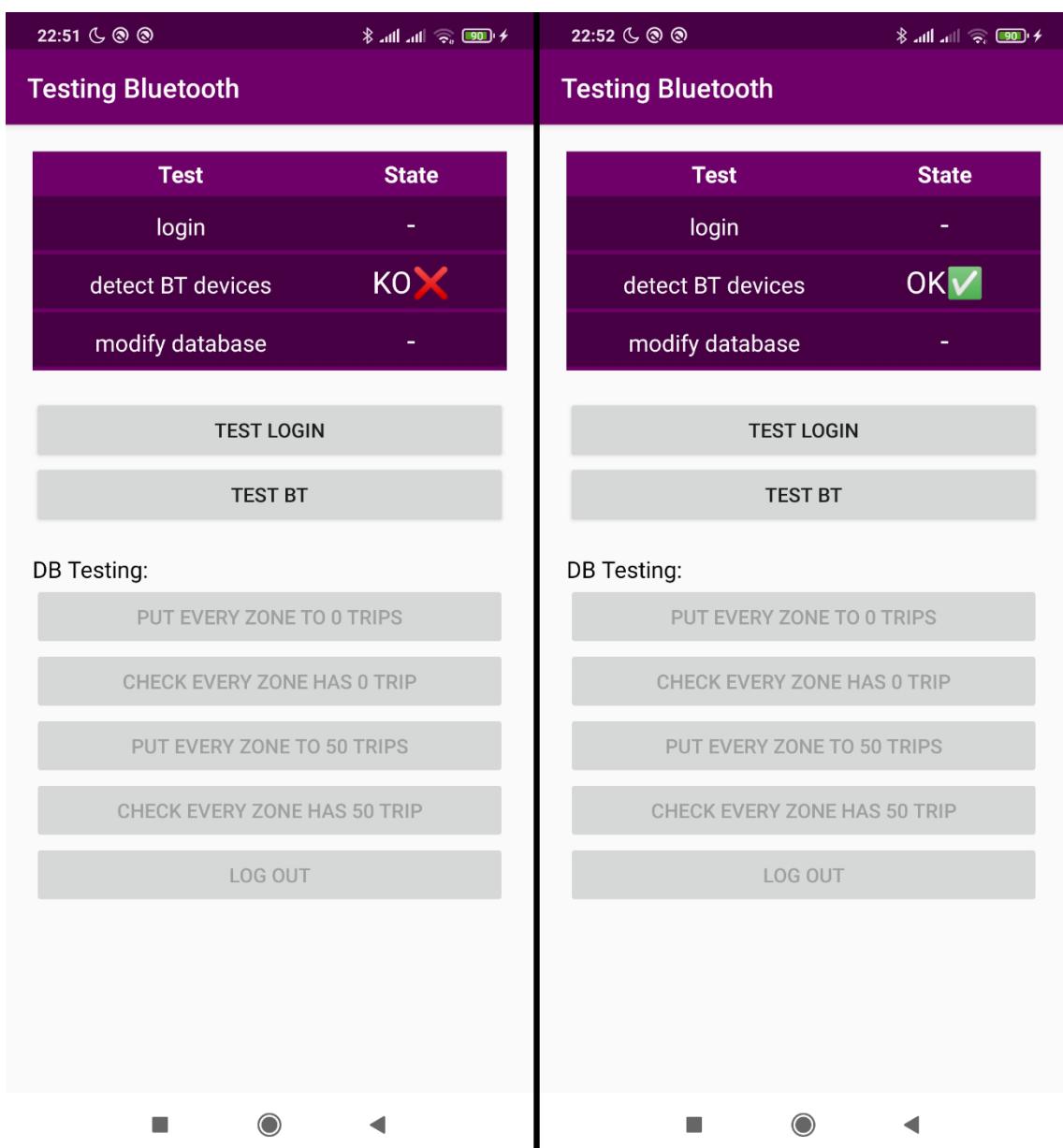


Figura 38 Prueba de bluetooth fallida y exitosa.

Al pulsar el botón de “Test login”, se iniciará sesión con unas credenciales reservadas para pruebas, y si el inicio de sesión es exitoso, se mostrará un “OK” verde en la tabla, tal y como muestra la Figura 39:

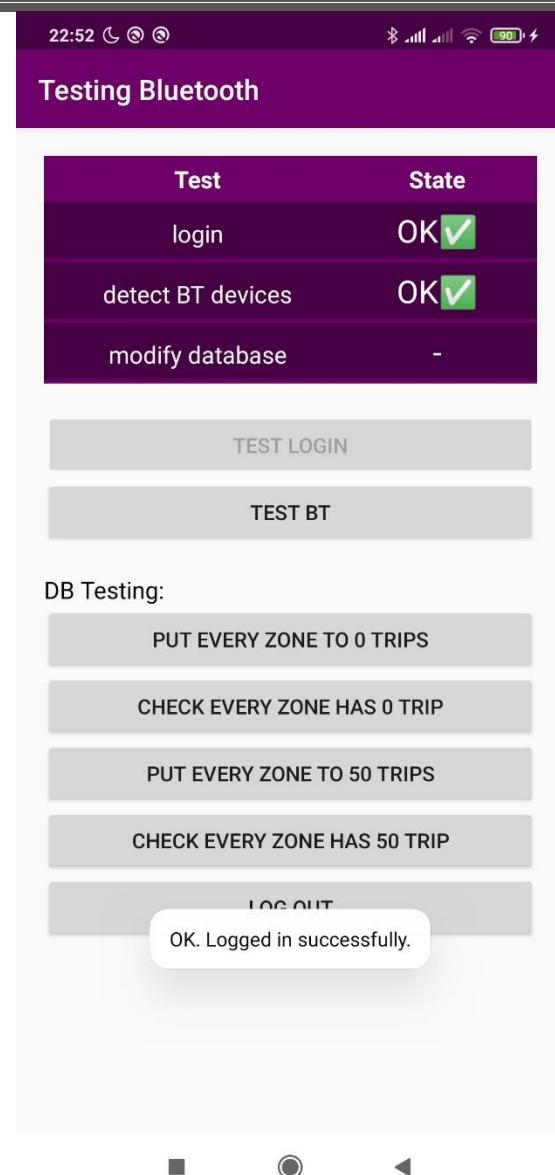


Figura 39 Test de inicio de sesión exitoso.

Una vez iniciada sesión, se desbloquean los botones de abajo, tal y como muestra la Figura 39, los cuales permiten testear diferentes funcionalidades sobre la base de datos de Google Firebase. Cada vez que se pulsa uno de ellos, se desencadenan una serie de operaciones, y si las mismas son exitosas, se muestra un “OK” verde en la tabla, tal y como muestra la Figura 40:

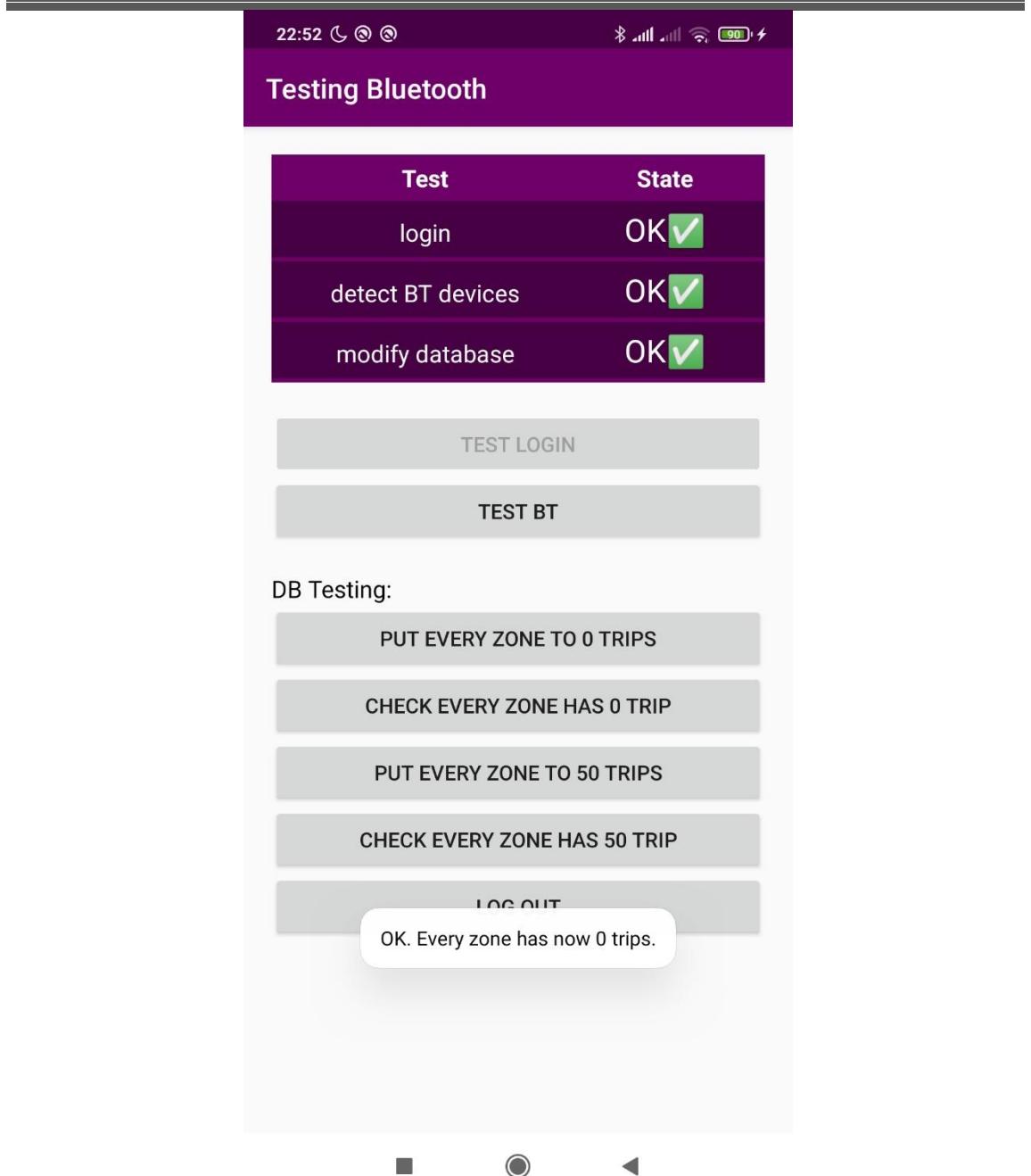


Figura 40 Pruebas de la base de datos de Google Firebase exitosas.

Una vez terminados los tests, y habiendo comprobado que el funcionamiento de la aplicación es correcto, podemos proceder a inhabilitar el botón “testApp” desde el código fuente, mostrado en la Figura 36, para que no aparezca en la pantalla principal. En el repositorio de GitHub, el botón de los test deberá estar siempre invisible, y solo se deberá hacer visible con el fin de probar la aplicación.

Cabe destacar que, si bien se ha barajado realizar los test en una base de datos duplicada, idéntica a la original y con los mismos datos, para garantizar la integridad de los datos de los usuarios ante un fallo en los tests, se ha descartado esa opción debido a la siguiente razón: Los test se realizan mediante una cuenta de usuario idéntica a las demás, sin ningún privilegio extra. En la consola de Google Firebase, podemos ver dicha cuenta, mostrada en la Figura 41:

Identificador	Proveedores	Fecha de creación
pruebatests@mail.com		14 jun 2022

Figura 41 Cuenta destinada a testing en la consola de Google Firebase.

Al ser una cuenta exactamente igual que las demás, es imposible modificar desde esa cuenta, los datos de otras cuentas de usuario. Por ejemplo, es imposible que estando identificado en esa cuenta se cometa un error que modifique los viajes de otros usuarios, o borre los mismos, puesto que la propia seguridad implantada de serie en la base de datos de Google evita que eso ocurra.

Debido a eso, y a que, tal y como indica la documentación de Google Firebase (Google, 2022), para hacer uso de las funciones de importación, exportación y duplicado de una base de datos, es necesario habilitar la facturación, no se han dedicado recursos en llevar a cabo este paso.

En resumen, los test simplemente simulan ser un usuario más modificando sus propios viajes, por lo que no podría alterar los viajes de otros usuarios, debido a que no tiene su contraseña. En caso de un fallo en los test, simplemente los viajes del usuario dedicado a testing se verán alterados, lo cual no supone un conflicto para la empresa contratante.

7.2 Pruebas de Integración y del Sistema

Para llevar a cabo las pruebas de integración, ejecutamos las diferentes funcionalidades del proyecto y anotamos el resultado que obtenemos en la Tabla 18, comparándolo con el que esperábamos.

Tabla 18 Pruebas de Integración y del Sistema.

Prueba	Resultado Esperado
Añadir un usuario no existente	El sistema posee un usuario más
	Resultado Obtenido
	El sistema efectivamente posee un usuario más
Prueba	Resultado Esperado
Añadir un usuario que ya existe	El sistema no posee un usuario más y se muestra un dialogo notificándolo
	Resultado Obtenido
	Efectivamente, el sistema no lo añade y notifica el error
Prueba	Resultado Esperado
Usuario pasajero compra bono	El sistema permanece añade a la base de datos un bono al usuario con la cantidad de viajes y el número de zonas que el usuario haya seleccionado.
	Resultado Obtenido
	El resultado obtenido es el esperado.
Prueba	Resultado Esperado
Usuario pasajero visualiza sus bonos	El sistema no cambia valores en su base de datos, simplemente consulta los bonos actuales del usuario y actualiza la base de datos.
	Resultado Obtenido
	El resultado obtenido es el esperado.
Prueba	Resultado Esperado
Usuario pasajero utiliza un bono en un autobús o tren	El sistema resta un viaje al usuario del número de zonas de las que sea el autobús o tren. Por ejemplo, si el autobús es de 4 zonas, el sistema resta un viaje de 4 zonas al usuario, y por tanto ahora le queda uno menos.
	Resultado Obtenido
	El resultado obtenido es el esperado.
Prueba	Resultado Esperado
Usuario pasajero intenta conectarse a un autobús o tren sin tener bonos.	El sistema no cambia valores en su base de datos, y simplemente le muestra al usuario que no puede conectarse a ese autobús o tren.
	Resultado Obtenido
	El resultado obtenido es el esperado.
Prueba	Resultado Esperado
Usuario conductor cambia el nombre de su vehículo	El sistema actualiza la base de datos y en la pantalla se muestra el nuevo nombre. Los cambios serán persistentes.
	Resultado Obtenido
	El resultado obtenido es el esperado.
Prueba	Resultado Esperado
Usuario conductor cambia el	El sistema actualiza la base de datos y en la pantalla se

número de zonas de su vehículo	muestra el nuevo número de zonas. Los cambios serán persistentes.
	Resultado Obtenido
	El resultado obtenido es el esperado.
Prueba	Resultado Esperado
Usuario conductor cambia la matrícula de su vehículo	El sistema actualiza la base de datos y en la pantalla se muestra la nueva matrícula. Los cambios serán persistentes.
	Resultado Obtenido
	El resultado obtenido es el esperado.

7.3 Pruebas de Usabilidad

A continuación, se enumeran las pruebas de usabilidad llevadas a cabo en el proyecto. Dichas pruebas se basan en la siguiente guía de evaluación heurística (Yusef Hassan Montero, 2003).

Criterios	¿Cumplido?
Generales	
¿Cuáles son los objetivos de las aplicaciones? ¿Son concretos y bien definidos? ¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?	SI
¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?	SI
¿Muestra de forma precisa y completa qué contenidos o servicios ofrece la aplicación? El diseño de la aplicación de inicio debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio.	SI
¿La estructura general de la aplicación está orientada al usuario? Las aplicaciones deben estructurarse pensando en el usuario, sus objetivos y necesidades. La estructura interna de la empresa u organización, cómo funciona o se organiza no interesan al usuario.	SI
¿El <i>look & feel</i> general se corresponde con los objetivos, características, contenidos y servicios de la aplicación? Ciertas combinaciones de colores ofrecen imágenes más o menos formales, serias o profesionales.	SI
¿Es coherente el diseño general de la aplicación? Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación.	SI
¿Es reconocible el diseño general de la aplicación? Cuanto más se parezca la aplicación al resto de aplicaciones, más fácil será de usar.	SI
Identidad e Información	
El título, ¿expresa realmente qué es la aplicación y qué servicios ofrece?	SI
Lenguaje y Redacción	
¿La aplicación habla el mismo lenguaje que sus usuarios? Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario.	SI
¿Emplea un lenguaje claro y conciso?	SI
¿Es amigable, familiar y cercano? Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario.	SI
¿1 párrafo = 1 idea? Cada párrafo es un objeto informativo. Trasmita ideas, mensajes...Se deben evitar párrafos vacíos o varios mensajes en un mismo párrafo.	SI
Rotulado	
Los rótulos, ¿son significativos? Ejemplo: evitar rótulos del tipo "haga clic aquí".	SI
¿Usa rótulos estándar? Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de..." .	SI

¿Usa un único sistema de organización, bien definido y claro? No se deben mezclar diferentes. Los sistemas de organización son: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas.	SI
¿Utiliza un sistema de rotulado controlado y preciso? Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de"	SI
Estructura y Navegación	
La estructura de organización y navegación, ¿Es la más adecuada? Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada,...	SI
En el caso de estructura jerárquica, ¿Mantiene un equilibrio entre Profundidad y Anchura?	SI
En menús de navegación, ¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística? No se deben superar los 7±2 elementos, ni los 2 o, como mucho, 3 términos por elemento.	SI
¿Se ha evitado la redundancia de información?	SI
¿Se ha controlado que no haya páginas "huérfanas"? Páginas huérfanas: que, aun siendo enlazadas desde otras páginas, éstas no enlacen con ninguna.	SI
Layout de la Página	
¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia? (como por ejemplo la zona central)	SI
¿Se ha evitado la sobrecarga informativa? Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Los grupos diferentes de objetos informativos de una página deben ser 7±2.	SI
¿Es una interfaz limpia, sin ruido visual?	SI
¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?	SI
¿Se hace un uso correcto del espacio visual de la página? Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia.	SI
¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página? (La jerarquía visual se utiliza para orientar al usuario)	SI
¿Se ha controlado la longitud de página? Se debe evitar en la medida de lo posible el <i>scrolling</i> . Si la página es muy extensa, se debe fraccionar.	SI
Accesibilidad (debería cubrirse con los test de Accesibilidad posteriores)	
¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?	SI
¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleadas facilitan la lectura?	SI
¿Existe un alto contraste entre el color de fuente y el fondo?	SI

¿Es compatible el sitio web con los diferentes dispositivos? ¿Se visualiza correctamente con diferentes resoluciones de pantalla? Se debe prestar atención a: <i>JScript, CSS, tablas, fuentes...</i>	SI
¿Puede el usuario disfrutar de todos los contenidos de la aplicación sin necesidad de tener que descargar e instalar <i>plugins</i> adicionales?	SI
¿Se ha controlado el peso de la página? Se deben optimizar las imágenes, controlar el tamaño del código <i>JScript</i> ...	SI
¿Se puede imprimir la página sin problemas? Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible.	SI
<i>Control y Retroalimentación</i>	
¿Tiene el usuario todo el control sobre el interfaz? Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos...	SI
¿Se informa constantemente al usuario acerca de lo que está pasando? Si el usuario tiene que esperar hasta que se termine una operación, se debe mostrar un mensaje indicándoselo y que debe esperar, con el tiempo de espera estimado o una barra de progreso.	SI
¿Se informa al usuario de lo que ha pasado? Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.	SI
Cuando se produce un error, ¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema? Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.	SI
¿Posee el usuario libertad para actuar? NO restringir la libertad del usuario: Uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación, no impida al usuario poder usar el botón derecho de su ratón...	SI
¿Se ha controlado el tiempo de respuesta? Esto tiene que ver con el peso de cada página (accesibilidad) y tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder. El tiempo máximo que esperará un usuario son 10 segundos	SI

En la Figura 42, se muestran ejemplos de la paleta de colores escogida para la aplicación:

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Desarrollo de las Pruebas

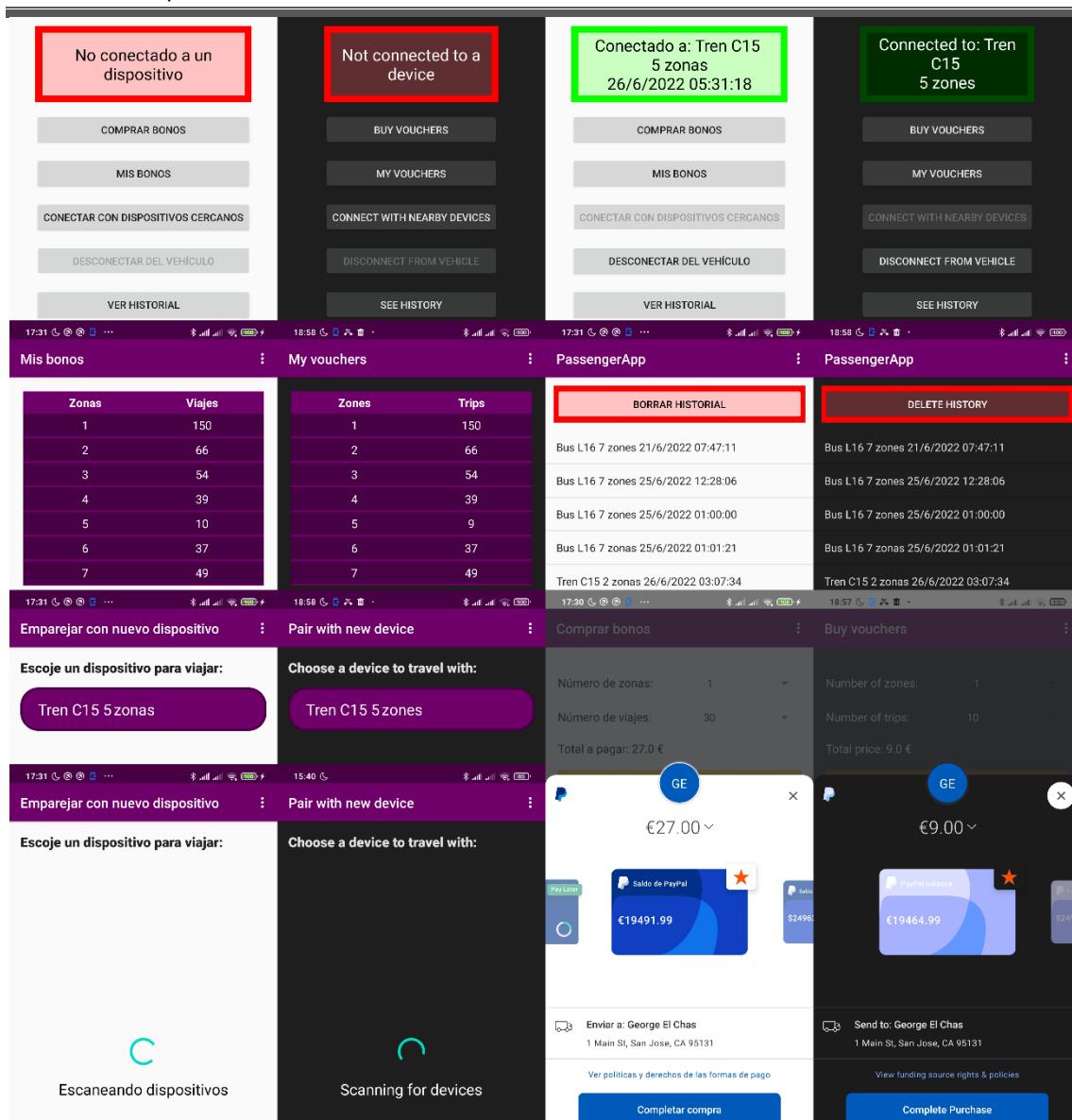


Figura 42 Ejemplo de paleta de colores de la aplicación PassengerApp.

Tal y como muestra la Figura 42, tanto en el tema oscuro como en el tema claro se muestran textos de colores claros sobre fondos oscuros, y viceversa . Según el siguiente sitio web de accesibilidad (Lea Verou, 2022), los colores utilizados cumplen los requisitos de accesibilidad, tal y como muestran la Figura 43:

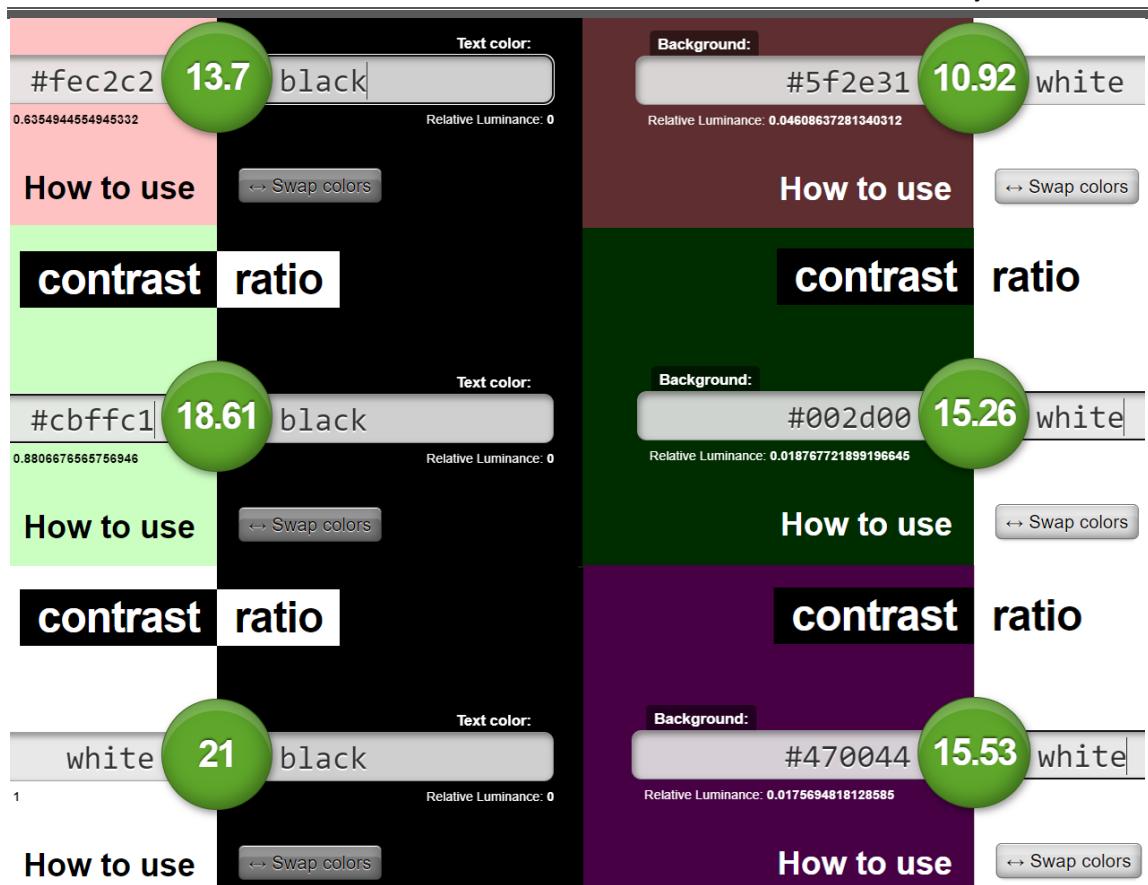


Figura 43 La paleta de colores cumple los requisitos de accesibilidad.

La Figura 43 muestra que, en todas las combinaciones de colores de la aplicación, se obtiene un resultado superior a la mitad de 21, el cual es el resultado de utilizar blanco y negro, obteniendo el contraste máximo.

Cabe destacar que si bien en la pantalla de pago de PayPal, cuando se utiliza el tema oscuro, se muestra una cifra en texto blanco sobre un azul claro, este código es de PayPal y no se puede modificar.

7.4 Pruebas de Rendimiento

Si bien no se han realizado pruebas automatizadas para medir el rendimiento, se han hecho pruebas manuales y los resultados han sido los esperados:

- Los tiempos de respuesta para consultas de la base de datos nunca han superado los dos segundos de duración.
- Los tiempos de detección de dispositivos bluetooth tienen una media de 3 segundos, lo cual es un resultado que permite el correcto uso de la aplicación.
- Se ha sido capaz de utilizar un móvil con la aplicación del vehículo conectado a la vez a 3 dispositivos móviles con aplicaciones del pasajero, y viceversa, y el resultado ha sido exitoso, sin interferencias y funcionando perfectamente. En la carpeta "Documentación App/Archivos adjuntos" se encuentran dos vídeos mostrando el funcionamiento con 4 dispositivos Android al mismo tiempo.

Capítulo 8. Manuales del Sistema

En esta sección se encuentran los manuales necesarios para instalar y ejecutar la aplicación, así como hacer uso de ella correctamente.

8.1 Manual de Instalación

Como ya se ha mencionado, el proyecto consta de dos aplicaciones Android. El proceso de instalación de ambas es completamente idéntico, por lo que solo se explicará para una de ellas, pues para la otra, el procedimiento es igual. Los pasos a seguir son los siguientes:

1. Tener instalada la versión más reciente de java, disponible en la siguiente referencia (Oracle, Página principal de Java, 2022).
2. Instalar los controladores USB de OEM, disponibles en ésta referencia (Google, Cómo instalar controladores USB de OEM, 2022), para nuestro dispositivo.
3. Descargar e instalar Android Studio (Google, Cómo instalar controladores USB de OEM, 2022).
4. Ejecutar la aplicación siguiendo el siguiente tutorial de Google (Google, Cómo ejecutar tu app, 2021).

Al instalar la aplicación desde un nuevo dispositivo Windows, habrá que realizar el siguiente paso adicional. En la pestaña “Gradle” de Android Studio, señalada en rojo en la Figura 44, haremos click sobre el icono del elefante de la izquierda, señalado a su vez en rojo.

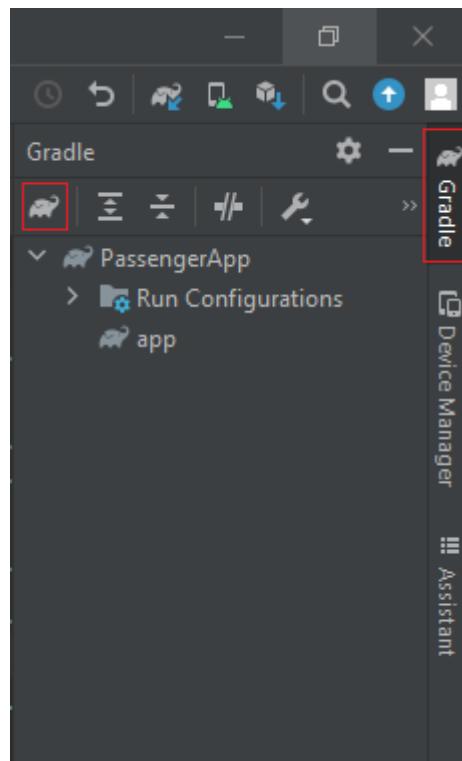


Figura 44 Pestaña gradle.

En la ventana emergente, teclearemos “gradle signInReport”, tal y como muestra la Figura 45, y presionaremos la tecla “intro”.

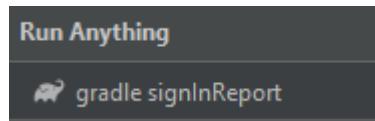


Figura 45 Comando “gradle signInReport”.

En la salida del comando en la consola, copiaremos el contenido del hash “SHA1”, el cual se muestra en la Figura 50, y será distinto en cada ordenador en el que instalamos la aplicación.

```
-----  
Variant: debugAndroidTest  
Config: debug  
Store: C:\Users\Al\.android\debug.keystore  
Alias: AndroidDebugKey  
MD5: B6:21:2C:96:99:B6:45:E2:19:26:4E:30:2A:C2:22:6A  
SHA1: 8E:73:74:81:21:DA:F6:75:3C:C6:02:6D:4E:20:A3:92:AA:DA:7B:B6  
SHA-256: 85:8C:69:0F:6F:C6:6F:50:9A:6A:C3:0F:C1:57:7C:A3:BA:8F:2B:E1  
Valid until: Wednesday, June 12, 2052  
-----
```

Figura 46 Contenido del hash “SHA1”.

Ese hash “SHA1” deberá ser añadido en el botón “Añadir huella digital”, mostrado en la Figura 47, en la configuración de la base de datos FireBase.

Configuración del SDK

¿Necesitas volver a configurar los SDK de Firebase en tu app? Revisa las instrucciones de configuración del SDK o descarga el archivo de configuración con las claves y los identificadores de tu app.

[Ver las instrucciones del SDK](#)

[!\[\]\(e33745601c56372ed8ebd264a174f6f8_img.jpg\) google-services.json](#)

ID de la app 

1:16142785235:android:32f33f30df0ee894b59226

Sobrenombre de la app

PassengerApp 

Nombre del paquete

com.example.passengerapp

Huellas digitales del certificado SHA 

Tipo 

8c:5e:fe:9a:f2:48:aa:33:a9:29:b5:f3:94:73:15:e2:ea:8a:46:f5	SHA-1
---	-------

8E:73:74:81:21:DA:F6:75:3C:C6:02:6D:4E:20:A3:92:AA:DA:7B:B6	SHA-1
---	-------

[Agregar huella digital](#)

Figura 47 Botón “Agregar huella digital” en la configuración de la base de datos de Firebase.

Debido a que solamente yo, el autor de este Trabajo de Fin de Grado, tengo las credenciales para acceder a la base de datos de Google Firebase (ya que está asociada a mi cuenta personal de Google), por favor, siéntanse libres de enviarme un email a la dirección uo258774@uniovi.es para añadir una determinada huella digital al proyecto, estaré encantado de hacerlo lo antes posible.

Dicho proceso deberá ser realizado cada vez que se instale la aplicación en un dispositivo Android desde un ordenador nuevo. De lo contrario, la base de datos de Firebase no funcionará en el proyecto. Esto se debe a que la aplicación no está en la tienda de Google (Play Store), se encuentra en estado de desarrollo, y por seguridad es necesario añadir el hash “SHA1” del proyecto en cada ordenador en el que se instale la aplicación.

Una vez hemos ejecutado la aplicación ya podremos utilizarla desde el dispositivo Android conectado por USB al ordenador, o bien desde el emulador de Android. Se recomienda encarecidamente utilizar un dispositivo Android real conectado por USB al ordenador, ya que la aplicación hace uso de las tecnologías Bluetooth y Localización, tecnologías de las que no dispone el emulador. Con un emulador tan solo se podrá hacer uso de la funcionalidad de comprar y visualizar bonos, ya que en ningún caso se podrá hacer uso tanto del bluetooth, como de la localización.

Para que la aplicación funcione es necesario tener el Bluetooth y la ubicación/localización del dispositivo activadas y darle permisos a la aplicación la primera vez que se ejecute.

8.2 Manual de Usuario

Como se ha mencionado, el proyecto consta de dos aplicaciones. Se dividirá por tanto el manual en 3 partes, las partes comunes de la aplicación, las partes de VehicleApp (la aplicación que irá instalada en el dispositivo Android del autobús o tren), y las partes de PassengerApp (la aplicación que irá instalada en los dispositivos Android de los pasajeros).

Cabe destacar que ambas aplicaciones detectan los ajustes tanto de idioma como de tema (oscuro o claro) del sistema operativo automáticamente, y en función de esos ajustes del teléfono, muestra la aplicación con tema oscuro o claro, en inglés o español, ya que está internacionalizada.

8.2.1 Partes comunes de la aplicación

En la siguiente sección se encontrarán las pantallas que son comunes a las dos aplicaciones, es decir, aquellas que son exactamente iguales en las dos aplicaciones.

Lo primero que debemos de hacer al ejecutar una aplicación por primera vez será iniciar sesión, en la pantalla mostrada en la Figura 48:

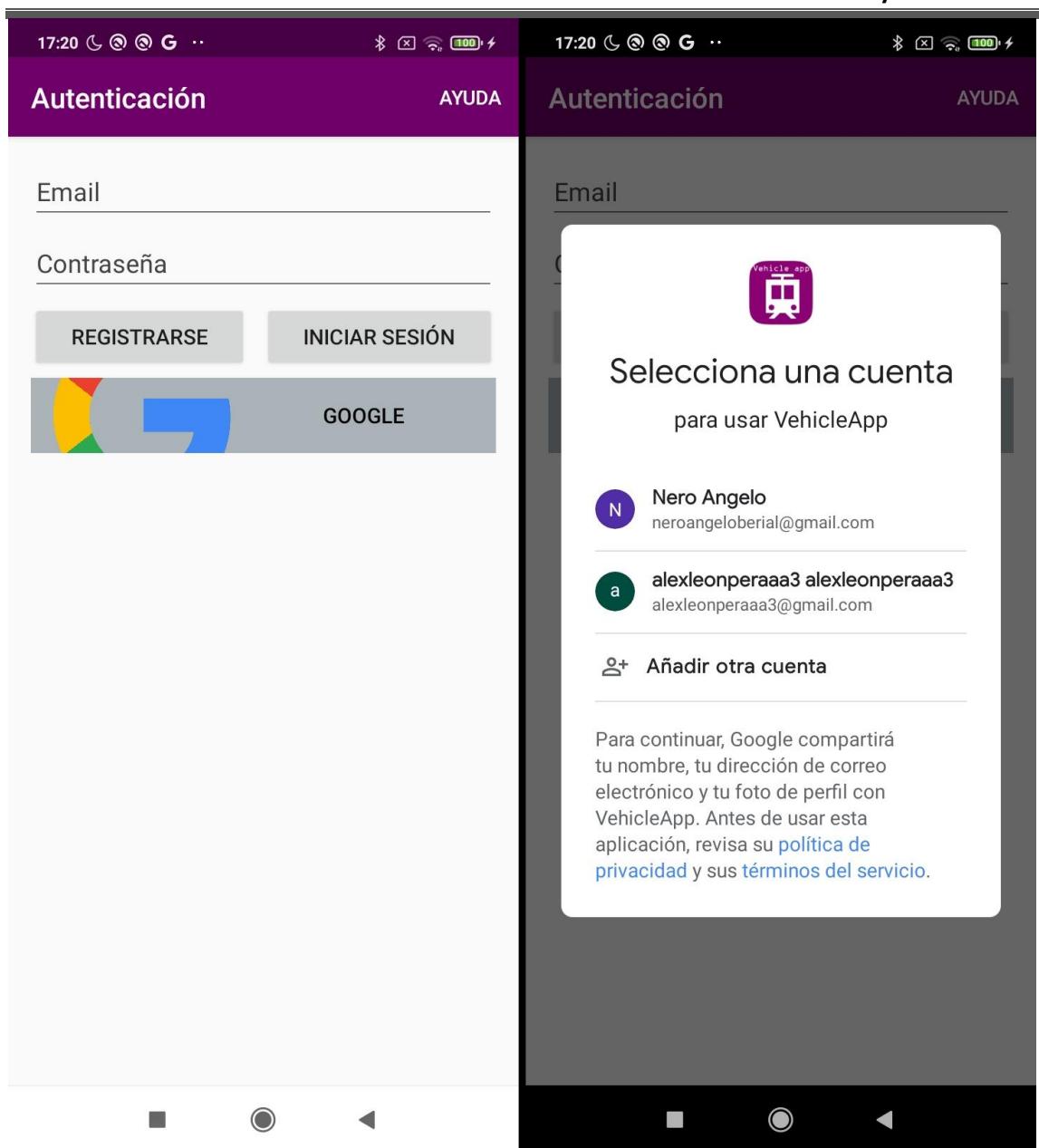


Figura 48 Pantallas comunes entre VehicleApp y PassengerApp que involucran la autenticación del usuario.

Si no tenemos una cuenta creada, podremos hacerlo bien rellenando un email y una contraseña, o bien utilizando una cuenta de Google existente en nuestro dispositivo. Una vez registrados podremos iniciar sesión como en cualquier otra aplicación, y no será necesario iniciar sesión todas las veces que abramos la aplicación, solamente la primera vez. (Si queremos cambiar de cuenta podremos cerrar sesión más adelante).

Una vez en la aplicación, si es la primera vez que la ejecutamos, deberemos asegurarnos de activar tanto el Bluetooth como la Ubicación y aceptar los permisos necesarios, tal y como se muestra en la Figura 49:

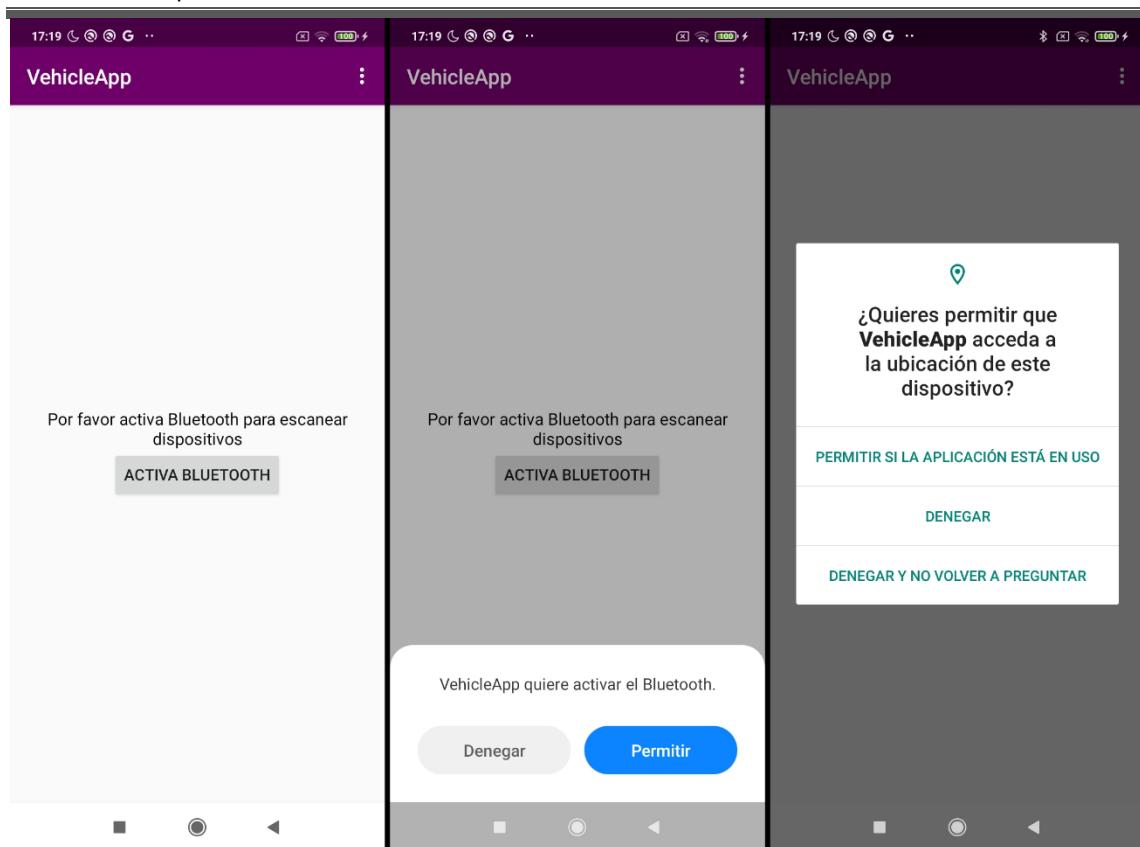


Figura 49 Pantallas comunes de VehicleApp y PassengerApp que piden permisos al usuario para utilizar el Bluetooth y la localización del dispositivo.

Una vez que le concedamos permisos a la aplicación, no volverá a solicitarlos.

8.2.2 VehicleApp

Ésta será la aplicación que irá instalada en los dispositivos Android de los vehículos (autobuses o trenes). Después de iniciar sesión y aceptar los permisos necesarios, nos aparecerá la interfaz mostrada en la Figura 50:

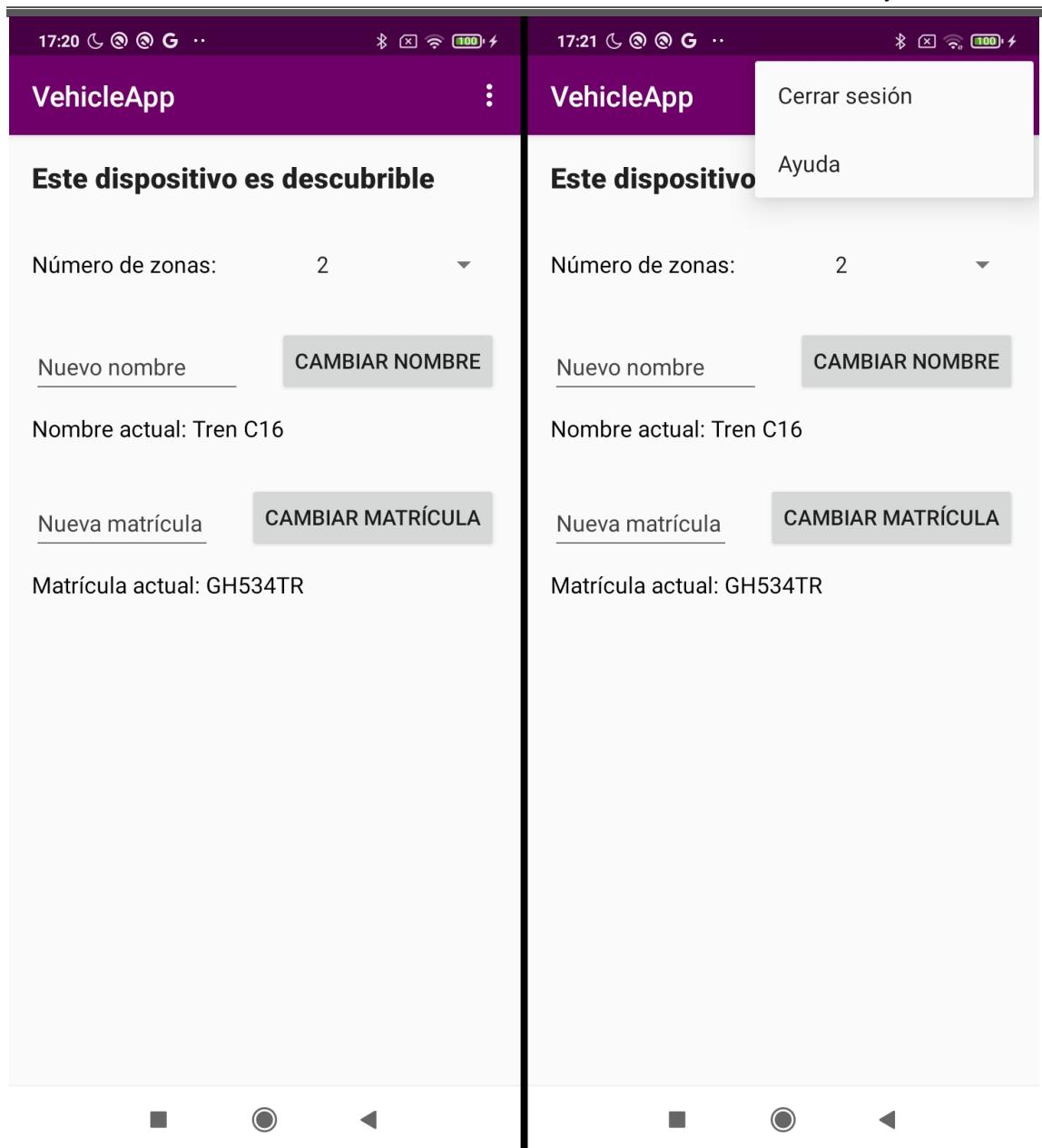


Figura 50 Pantalla principal de VehicleApp.

Mediante esta interfaz, el conductor del autobús o maquinista podrá seleccionar el número de zonas de su vehículo, el nombre del mismo, y su matrícula o número de identificación, en caso de existir. Al utilizar la aplicación por primera vez, no existe ningún nombre y habrá que escribir uno y darle al botón “Cambiar nombre”. La app se reiniciará automáticamente una vez pulsemos el botón “Cambiar”.

El menú de arriba a la izquierda nos permite tanto cerrar sesión, como mostrar la ayuda del usuario. Al cerrar sesión, nos llevará al menú de inicio de sesión, y al seleccionar la ayuda, nos mostrará el manual de usuario de la aplicación, tal y como muestra la Figura 51:

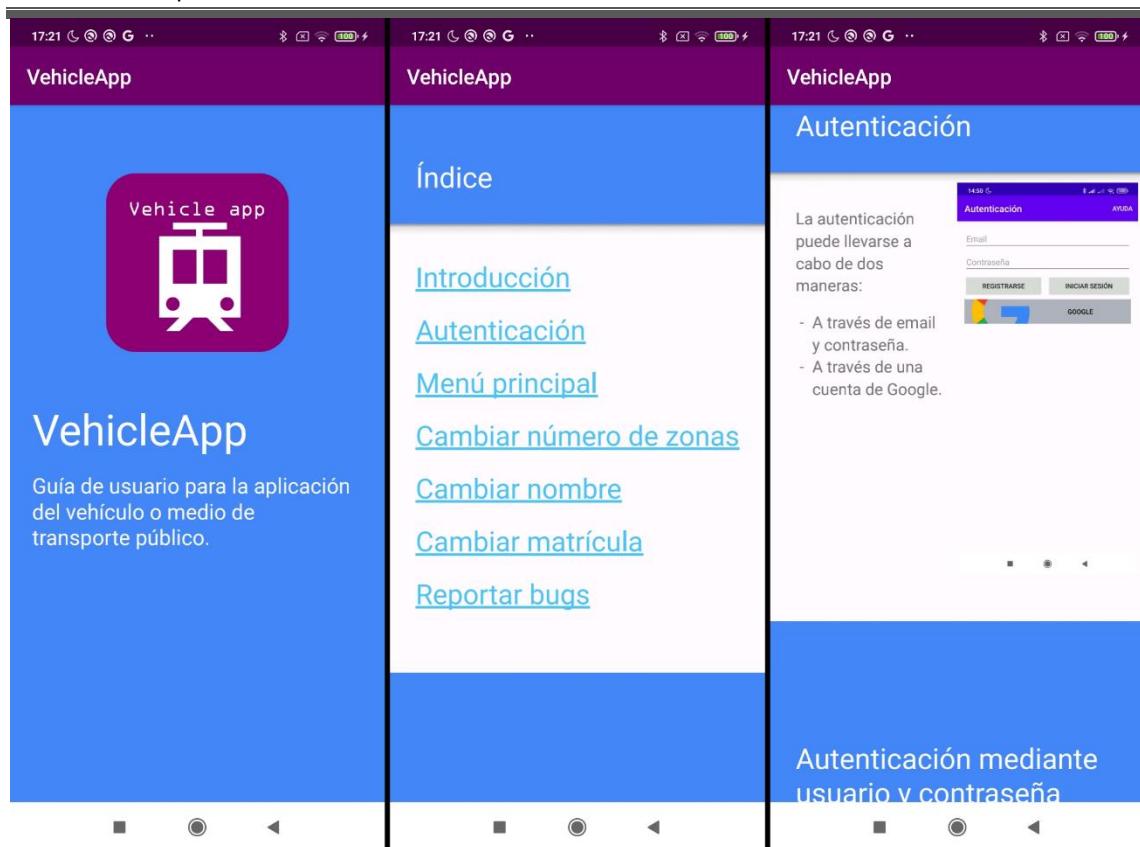


Figura 51 Manual de usuario de VehicleApp.

8.2.3 PassengerApp

PassengerApp será la aplicación que instalarán los pasajeros en sus dispositivos móviles.

Al iniciar sesión y aceptar los permisos se mostrará la pantalla de la Figura 52:



Figura 52 Pantalla principal de PassengerApp.

Esta pantalla nos muestra la interfaz principal. El recuadro rojo nos indica que no estamos haciendo uso de ningún bono (no nos hemos subido a ningún autobús o tren).

El primer botón (Comprar bonos), nos permite comprar bonos de transporte público, como se muestra en la Figura 53 y en la Figura 54:

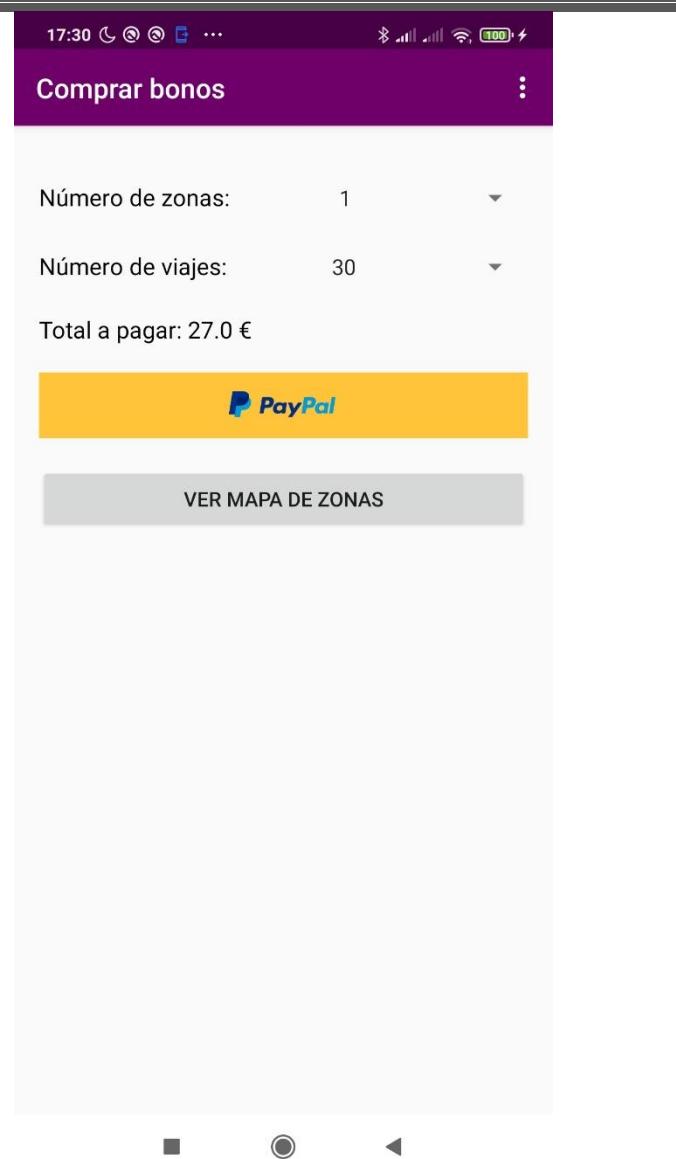


Figura 53 Pantalla de “comprar bonos” de PassengerApp.

Podemos seleccionar el número de zonas y el número de viajes. En el botón “PayPal” podemos hacer uso de nuestra cuenta de PayPal para comprar los bonos, tal y como se muestra en la Figura 54:

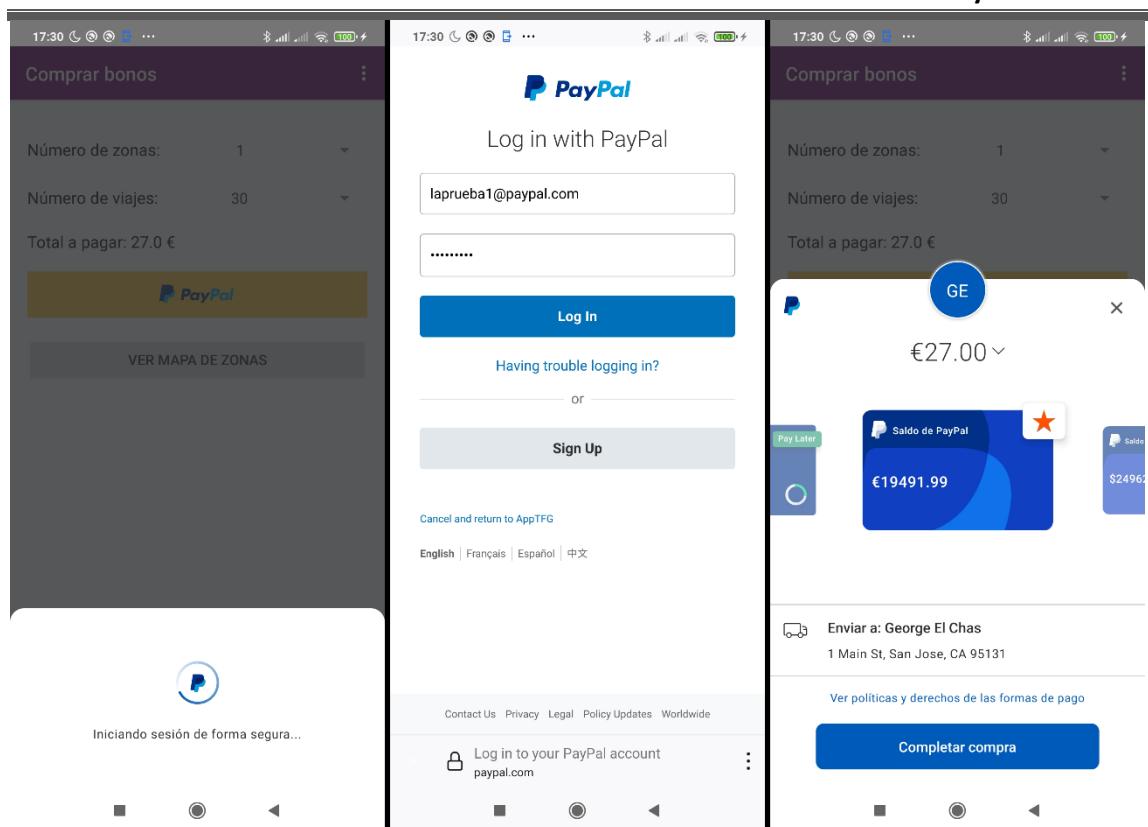


Figura 54 Uso de PayPal para comprar 30 viajes de 1 zona.

Desde la ventana de “Comprar bonos”, pulsando el botón “Ver mapa de zonas”, también podemos ver el mapa de zonas de la CTA, mostrado en la Figura 55:

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Manuales del Sistema

The figure consists of three screenshots of the PassengerApp mobile application. The first screenshot shows the app's logo and a placeholder text 'Mapa de zonas'. The second screenshot shows a map of Asturias divided into numbered zones (1-30). The third screenshot shows a table of numbers corresponding to the map, with a red box highlighting the number 12.

PassengerApp

El siguiente mapa muestra la división de zonas de Asturias.

PassengerApp

Para calcular el número de zonas que es necesario para ir de una parte de Asturias a otra, se deben mirar los números en el mapa y consultar en la tabla cuántas zonas hay entre esos números. Por ejemplo, para ir desde Oviedo (ciudad 12) hasta Avilés (ciudad 2) son necesarias 3 zonas:

	1	2
1	1	
2	2	1
3	3	2
4	4	4
5	5	4
6	6	4
7	7	2
8	8	3
9	9	4
10	10	4
11	11	4
12	12	3
13	13	3
14	14	3
15	15	2
16	16	2
17	17	2
18	18	2
19	19	2
20	20	2
21	21	2
22	22	2
23	23	2
24	24	2
25	25	2
26	26	2
27	27	2
28	28	2
29	29	2
30	30	2

El siguiente mapa muestra la división de zonas de Asturias.

El siguiente mapa de zonas muestra cuántas zonas son necesarias para viajar de un sitio a otro de Asturias. El funcionamiento de las zonas se explica a continuación.

Mapa de zonas

Figura 55 Mapa de zonas de la CTA en Asturias.

Una vez comprados los bonos, los podremos visualizar en la pantalla a la que nos lleva el botón de "Mis bonos", mostrada en la Figura 56:

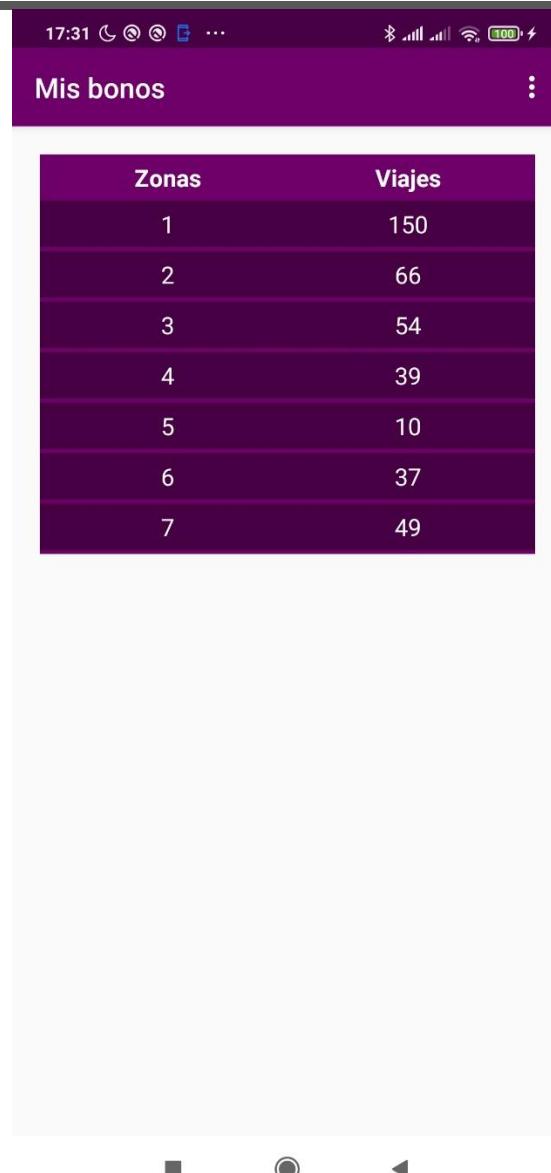


Figura 56 Pantalla de “Mis bonos” de PassengerApp.

En esta pantalla podremos ver de qué bonos disponemos. A la izquierda se muestra el número de zonas y a la derecha cuántos viajes tenemos para ese número de zonas.

El tercer botón del menú principal “Conectar con dispositivos cercanos”, nos permite conectarnos a vehículos para gastar nuestros bonos, como se muestra en la Figura 57:

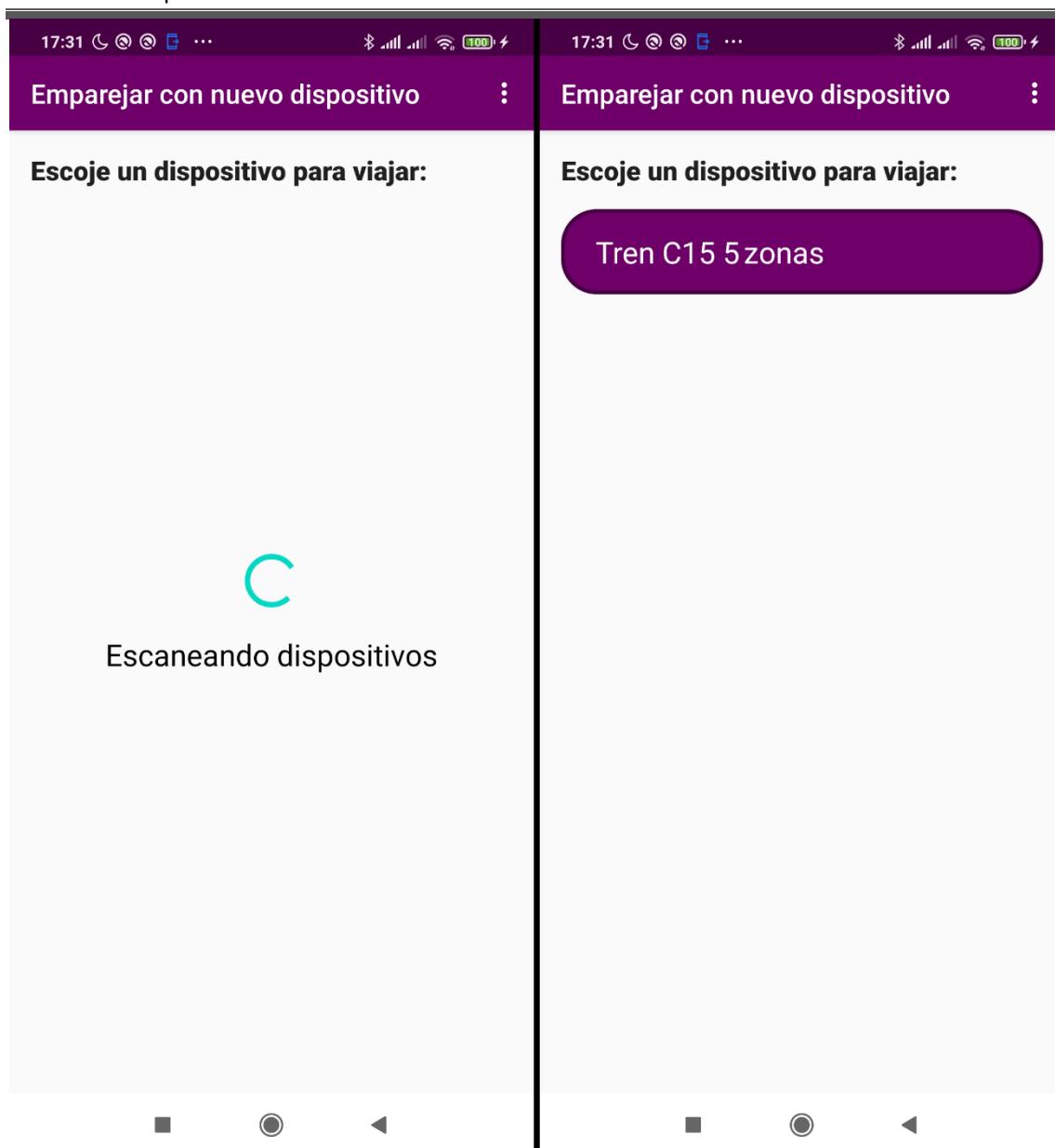


Figura 57 Aplicación del pasajero buscando un vehículo.

En esta pantalla se escaneará en busca de dispositivos. Cabe destacar que solamente aparecerán los dispositivos que estén ejecutando la aplicación “VehicleApp”, por lo tanto, no nos aparecerán aquí otros dispositivos Bluetooth como auriculares, sino que solamente aparecerán autobuses o trenes.

Si seleccionamos “Tren C15 5 zonas” nos conectará a ese tren, gastando un bono de 2 zonas, tal y como se muestra en la Figura 58:

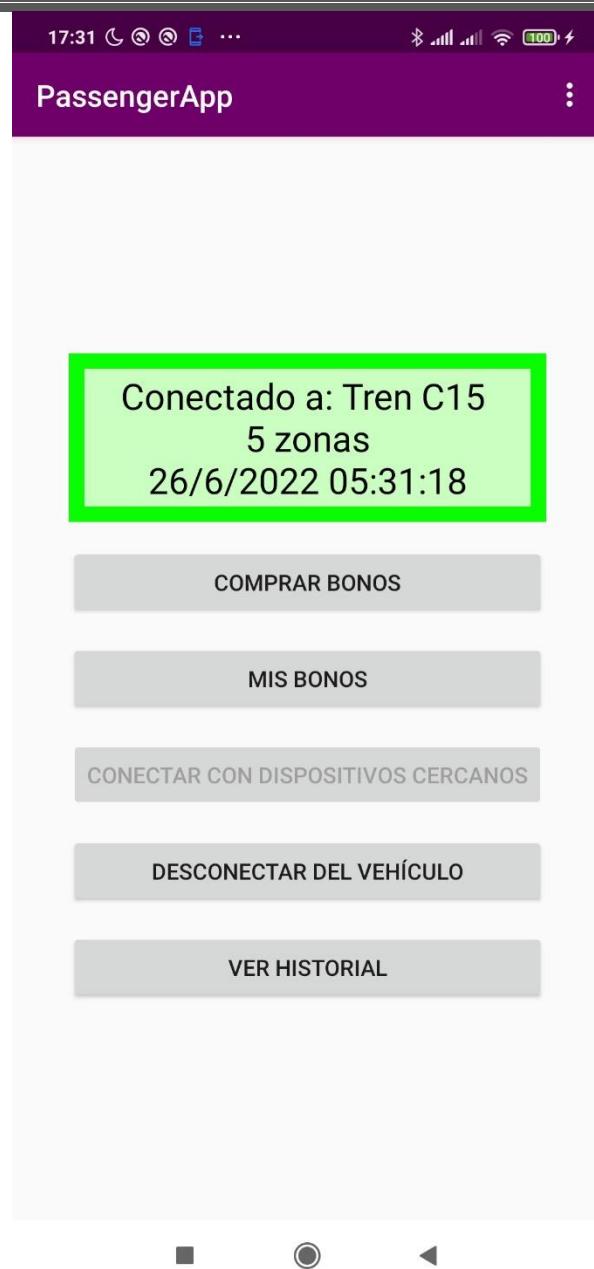


Figura 58 Aplicación del pasajero conectada a un autobús.

Como podemos observar en la Figura 58, ahora en la pantalla principal se muestra el recuadro en verde con el vehículo al que nos hemos conectado, y a qué hora. Esto se lo deberán enseñar los pasajeros al revisor del autobús o tren en caso de que se lo requiera.

También tenemos a nuestro alcance un historial, mostrado en la Figura 59, el cual también podremos mostrar al revisor, y lo podremos eliminar cuando queramos.

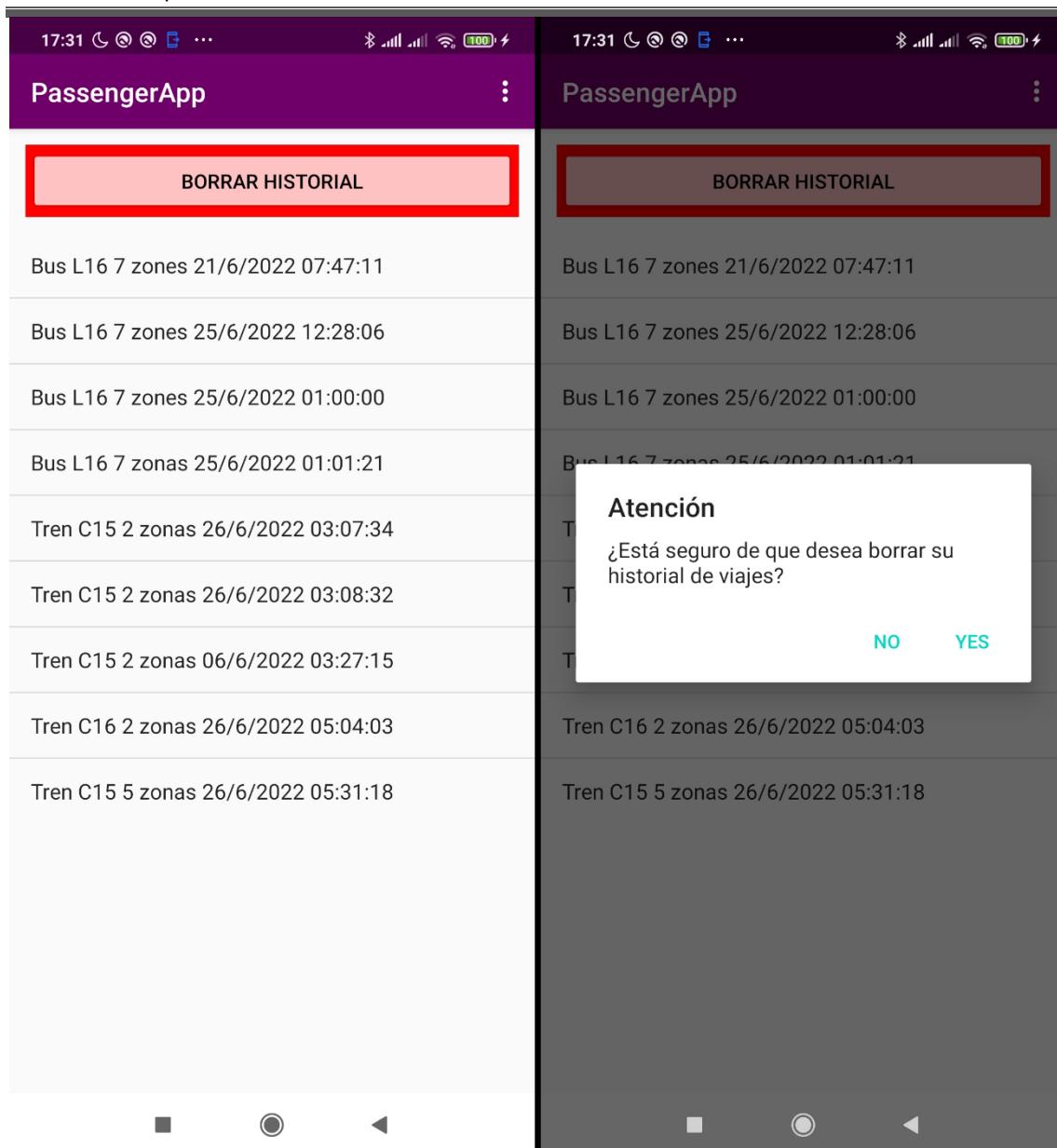


Figura 59 Historial de viajes de la aplicación del pasajero.

Si ahora entramos en la pantalla de “Ver bonos”, vemos que ahora tenemos un viaje menos de 2 zonas, tal y como se muestra en la Figura 60:

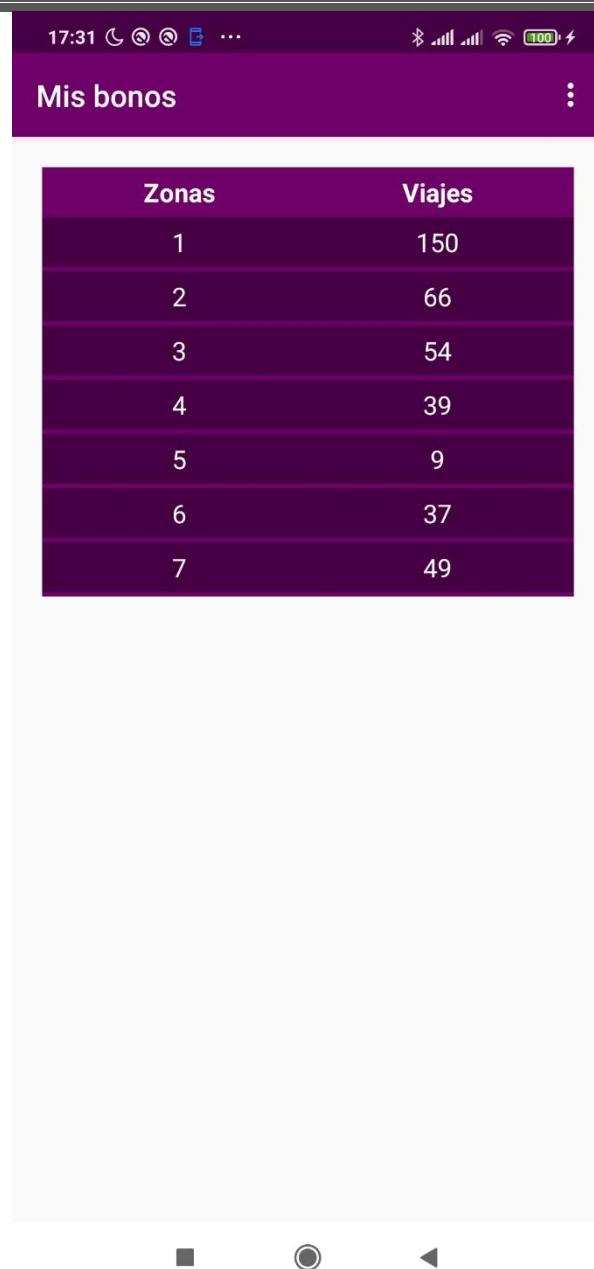


Figura 60 Bonos del usuario.

Como podemos ver en la Figura 60, tenemos 9 viajes, mientras que antes teníamos 10.

El dispositivo del pasajero seguirá conectado hasta el momento en el que se encuentre lo suficientemente lejos del autobús o tren como para perder la conexión Bluetooth, o bien nos desconectemos manualmente desde el botón “Desconectar del vehículo”, mostrado en la Figura 61:

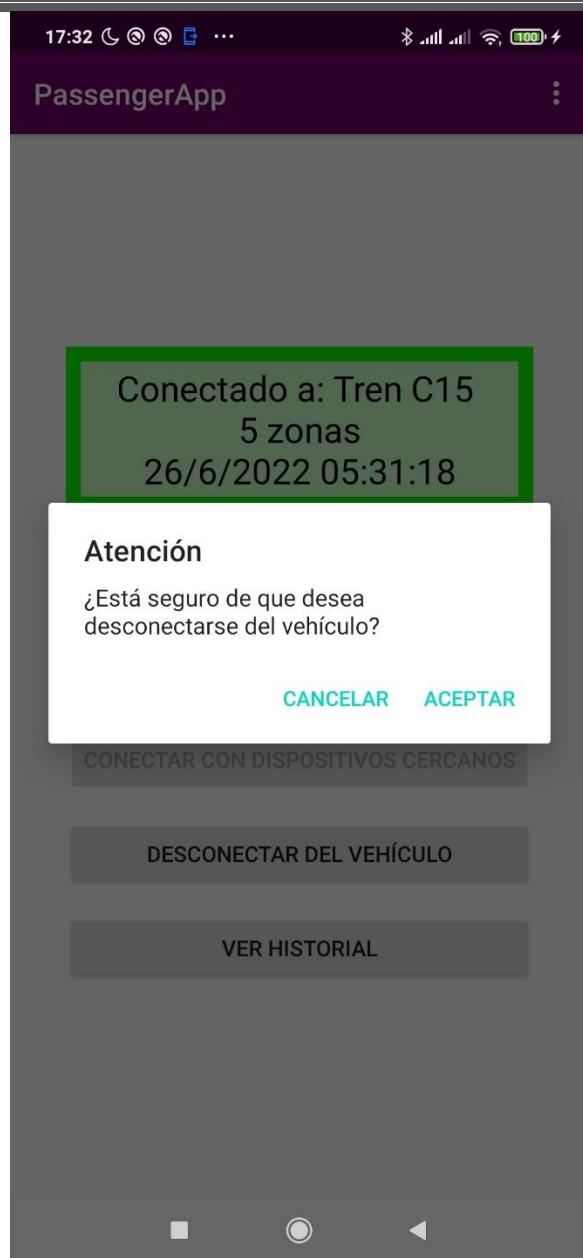


Figura 61 Pasajero desconectándose manualmente del vehículo.

En ese momento, se nos volverá a mostrar la pantalla principal con el recuadro en rojo, tal y como muestra la Figura 62:

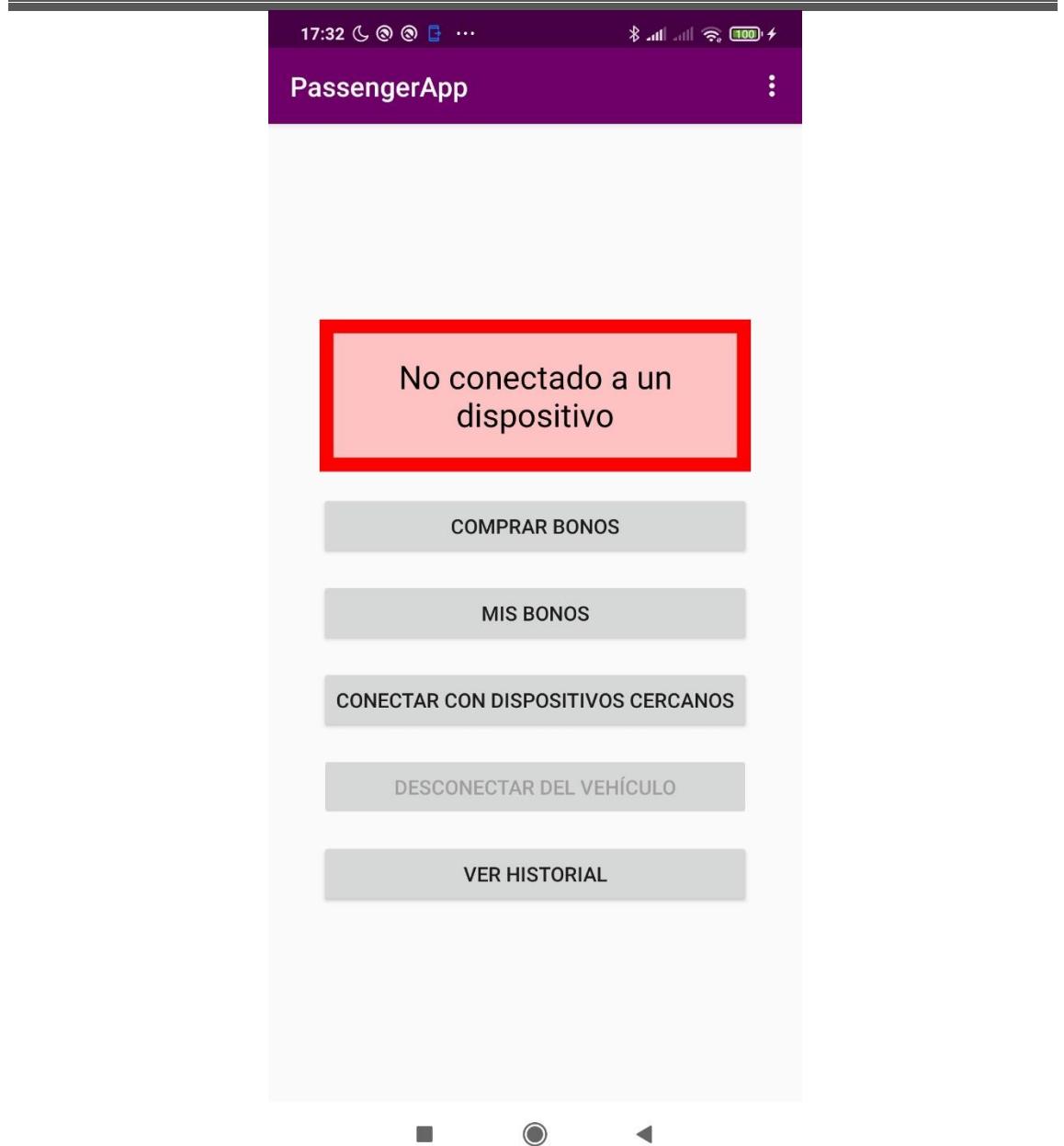


Figura 62 Dispositivo del pasajero desconectado.

Como se ha comentado al principio del apartado 8.2, la aplicación tiene un tema oscuro (para facilitar la accesibilidad a ciertos grupos de usuarios), y está internacionalizada.

Para demostrar lo dicho, en la Figura 63, Figura 64 y Figura 65, se muestran unas capturas de pantalla de la aplicación del pasajero en idioma inglés y tema oscuro. No se mostrará la aplicación entera de nuevo pues sería redundante, solo se muestran algunas de las pantallas más relevantes:

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Manuales del Sistema

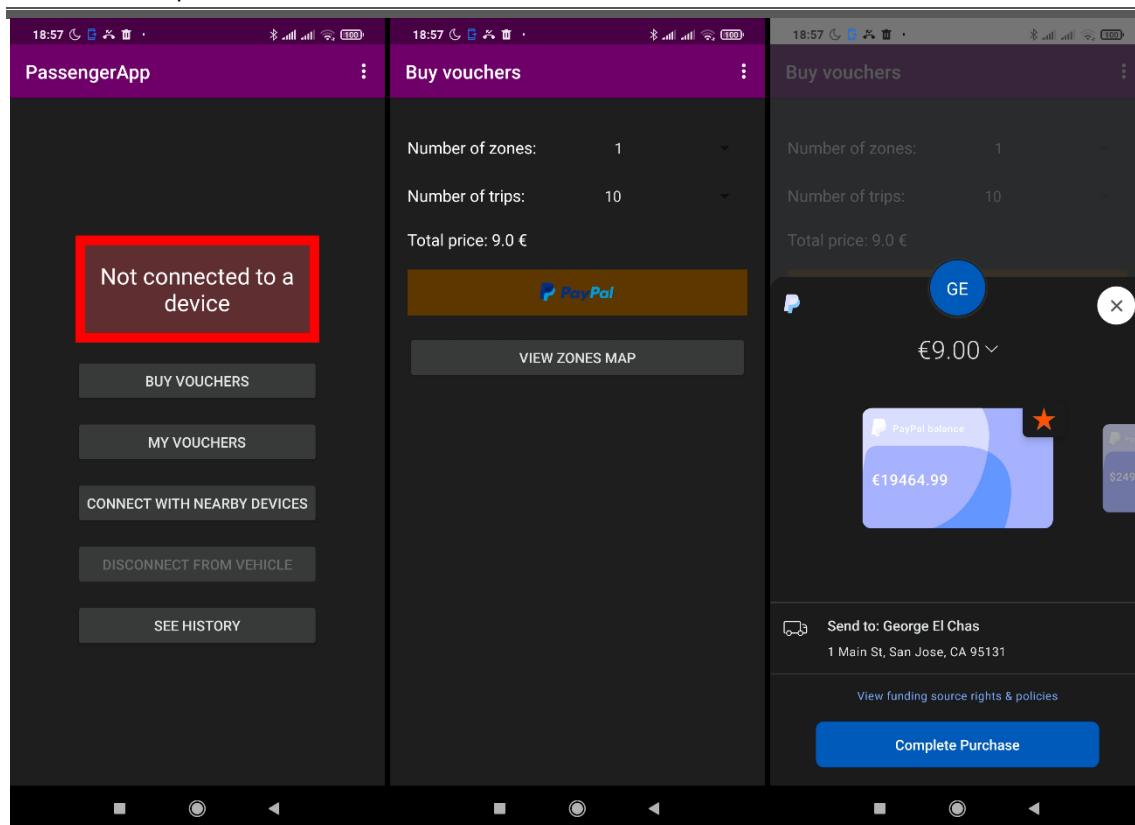


Figura 63 Pantallas más relevantes de la aplicación del pasajero en inglés y con el tema oscuro.

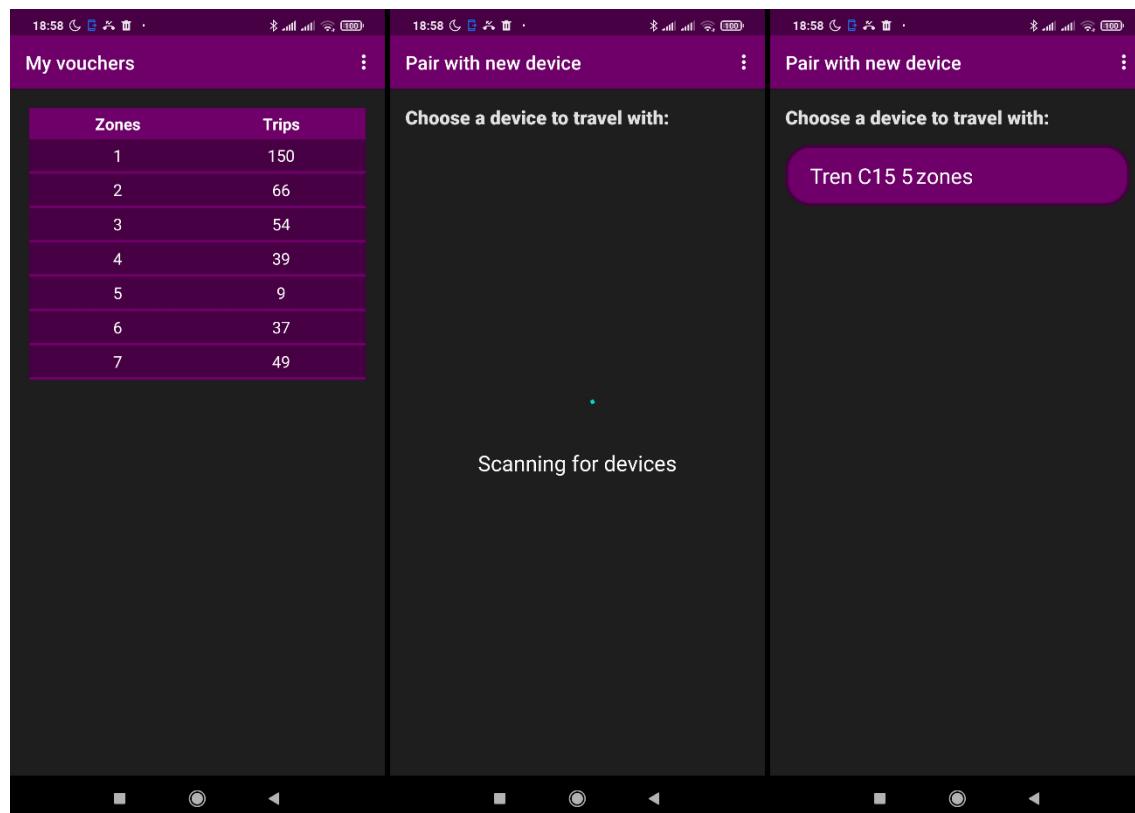


Figura 64 Pantallas más relevantes de la aplicación del pasajero en inglés y con el tema oscuro (Parte 2).

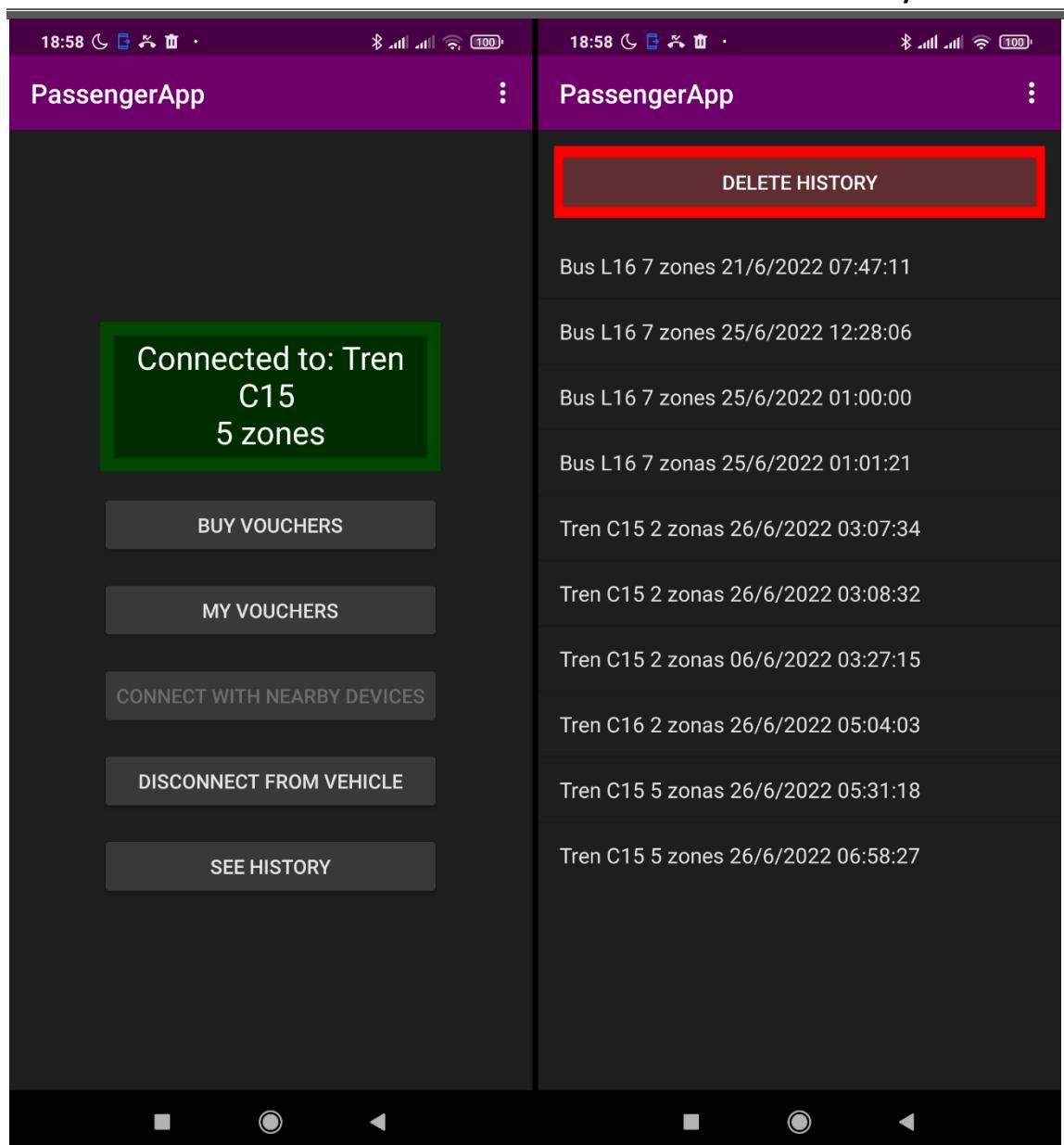


Figura 65 Historial de la aplicación del pasajero en idioma inglés y tema oscuro.

8.3 Manual del Programador

En este se describen los aspectos que pueden ayudar a otros programadores a trabajar con nuestra aplicación. Se dividirá este punto en dos apartados, uno para cada aplicación desarrollada.

8.3.1 Manual del programador para PassengerApp

Si bien las dos aplicaciones se dividen en 4 paquetes principales (bluetooth, protocol, scan y appTesting) más las clases de la raíz, el esquema de la aplicación del pasajero es el mostrado en la Figura 66:

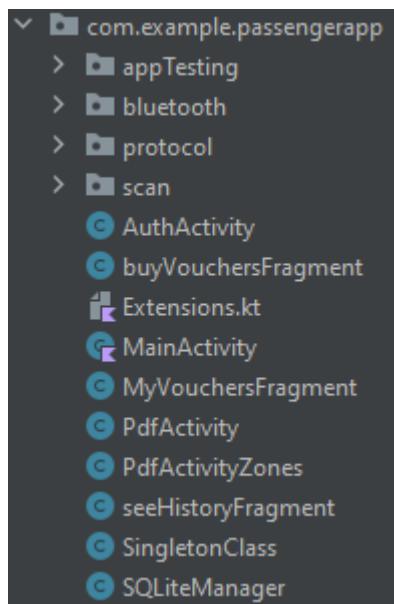


Figura 66 Paquetes principales PassengerApp.

En el paquete “appTesting”, mostrado en la Figura 67, se encuentran tanto la actividad responsable de los test de la aplicación, como su fragmento, donde se encuentra la interfaz de usuario de los mismos. Toda la información sobre los tests se ha detallado en el apartado 7.1.

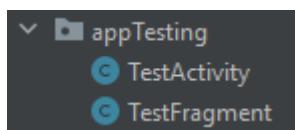


Figura 67 Paquete appTesting de PassengerApp.

En el paquete bluetooth, mostrado en la Figura 68, está todo lo relacionado con el módulo de bluetooth de las aplicaciones. Funcionan de la misma manera y con las mismas clases en ambas aplicaciones. No se deberían tener que modificar las clases del paquete bluetooth, excepto la clase BluetoothServer.kt, clase principal donde se almacena la lógica principal del módulo bluetooth de la aplicación.

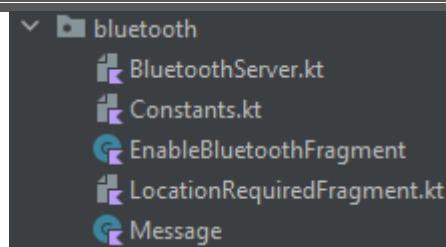


Figura 68 Paquete bluetooth de PassengerApp.

En el módulo protocol, mostrado en la Figura 69, se encuentran clases necesarias para el intercambio de información entre los dispositivos. Para desarrollar no debería ser necesario tocar código de ninguna de esas clases.

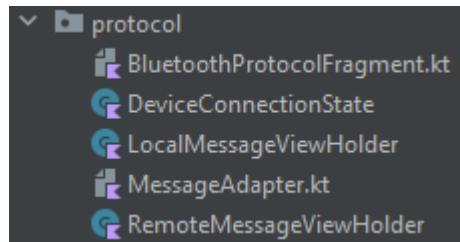


Figura 69 Módulo protocol de PassengerApp.

El tercer paquete, scan, contiene las clases necesarias para manejar la visibilidad del dispositivo y para detectar otros dispositivos. Contiene las mismas clases en las dos aplicaciones (menos la clase DeviceScanFragTest.kt, la cual solo está en PassengerApp), y se muestran en la Figura 70. La clase DeviceScanFragTest.kt es un duplicado de la clase DeviceScanFragment.kt, modificado ligeramente para hacer uso de él en las clases de los test, mostradas en la Figura 67.

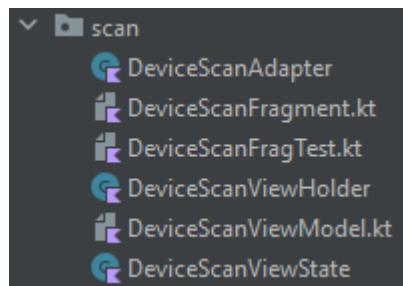


Figura 70 Paquete scan de PassengerApp.

Respecto a la autenticación de los usuarios, en las dos aplicaciones se encuentra en la raíz del paquete com.example.passengerapp, mostrado en la Figura 71:

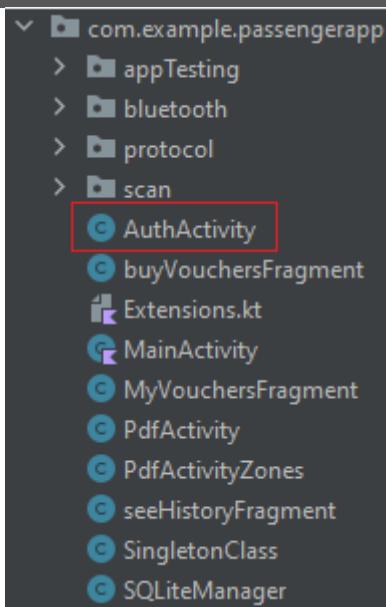


Figura 71 Raíz del proyecto PassengerApp.

En la clase `AuthActivity` (mostrada en la Figura 71), se encuentra toda la lógica necesaria para comunicarse con la base de datos de Google Firebase, para autenticar a los usuarios. Es igual tanto en `PassengerApp` como en `VehicleApp`. En las partes del código donde se hacen conexiones con la base de datos se han puesto comentarios especificando qué y cómo lo hacen.

La clase `Extensions.kt` simplemente almacena los tres tipos de visibilidad que puede tener un objeto en la interfaz de usuario:

- `Visible`: el elemento está presente en la interfaz, y el usuario puede verlo.
- `Invisible`: el elemento ocupa espacio en la interfaz, pero el usuario no puede verlo porque es transparente.
- `Gone`: (en español significa “desaparecido”). El elemento no existe en la interfaz y no ocupa ningún espacio en pantalla. En ningún caso el usuario puede notar la existencia de los elementos marcados como “gone”.

En la clase `MainActivity` se encuentra el código necesario para inicializar la aplicación y lanzar la primera ventana de la misma, es decir, la pantalla de inicio de sesión. En el caso de que el usuario ya esté autenticado, se salta a la pantalla del menú principal.

En las clases `buyVouchersFragment` y `MyVouchersFragment` se encuentra la funcionalidad encargada con comprar bonos de transporte público, y visualizarlos, respectivamente.

En las clases `PdfActivity` y `PdfActivityZones` se encuentra la funcionalidad encargada de mostrar al usuario el archivo PDF de Ayuda de la aplicación, y el PDF de ayuda para saber cuántas zonas necesita un usuario, respectivamente.

En la clase `seeHistoryFragment` se encuentra la funcionalidad referente al historial del usuario, donde tiene almacenada la información de sus últimos viajes (nombre del vehículo, número de zonas, y fecha). Los viajes, están almacenados en una base de datos local del dispositivo móvil del usuario. La base de datos se encuentra en la clase `SQLiteManager`.

Por último, la clase SingletonClass, común a ambas aplicaciones, se trata de la clase única de un patrón de diseño llamado Singleton, el cual consiste en almacenar todos los atributos de la aplicación que necesitan ser accedidos desde muchas partes del código, para que puedan ser compartidos desde distintas clases de manera sencilla.

El patrón de diseño singleton es sencillo, en resumen, permite tener unos atributos a los que se puede acceder desde cualquier sitio de la aplicación sin importar donde. Esto se consigue teniendo una sola instancia de un objeto, que es posible modificar desde todos los puntos de la aplicación.

Por último, y referente a la interfaz de usuario, los archivos XML que la definen se encuentran dentro de la carpeta res (del inglés resource), en la carpeta layout. En la carpeta navigation se encuentra un mapa de navegación entre las diferentes pantallas de la aplicación. Por ejemplo, para PassengerApp es el mostrado en la Figura 72:

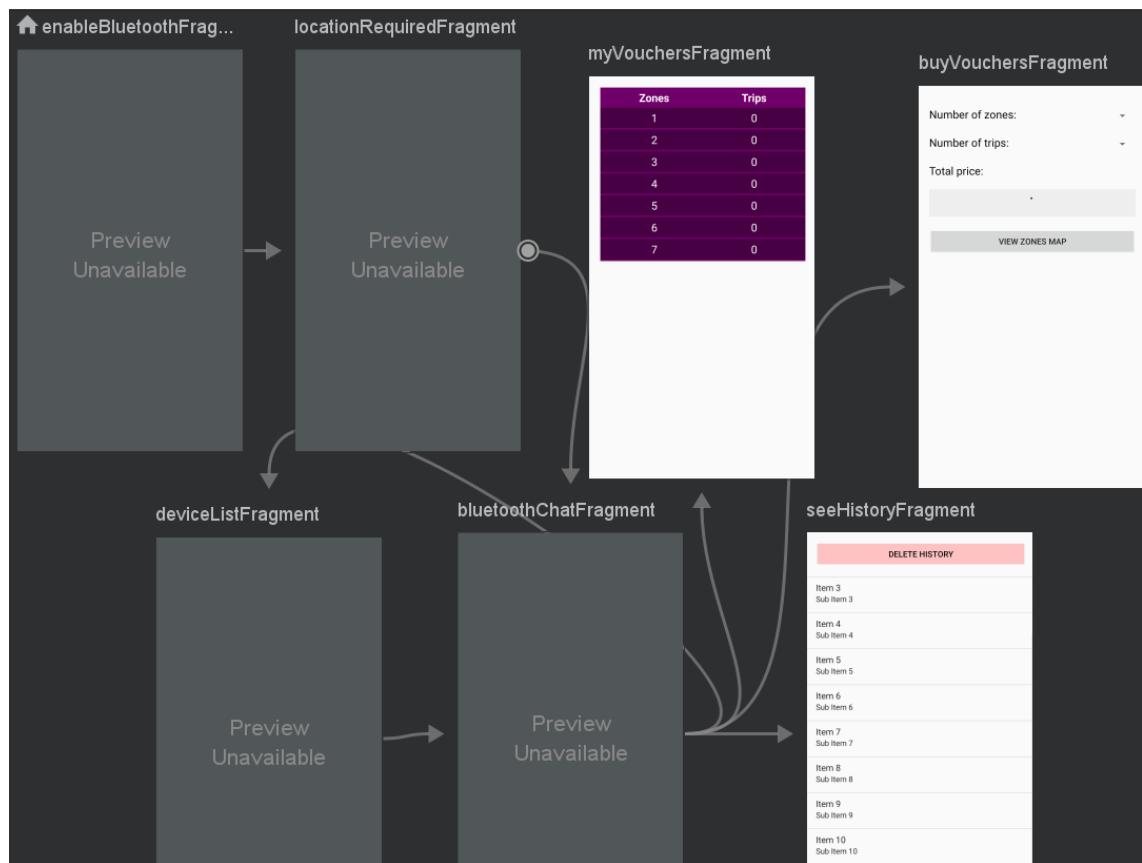


Figura 72 Diagrama de pantallas de PassengerApp.

Se pueden añadir nuevas pantallas al proyecto desde el archivo nav_graph.xml.

8.3.2 Manual del programador para VehicleApp

El esquema de clases de VehicleApp es el mostrado en la Figura 71:

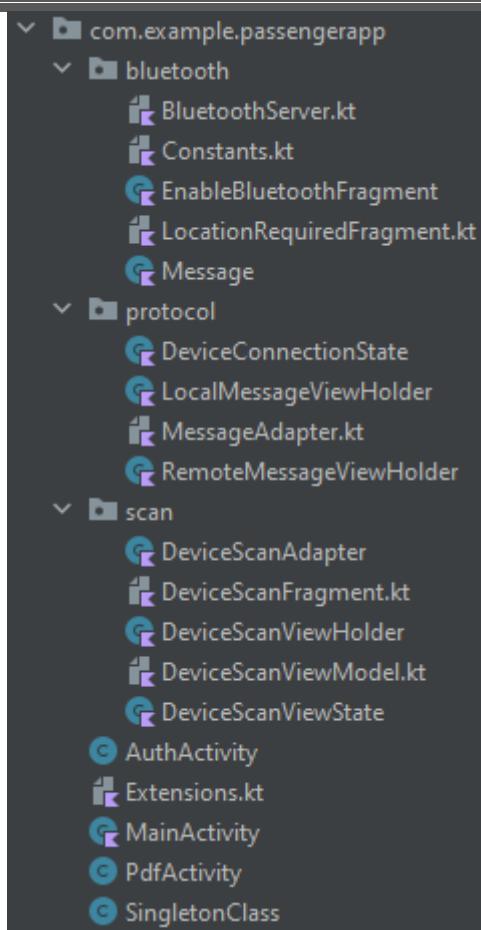


Figura 73 Diagrama de clases de VehicleApp.

Si bien todas las clases presentes en la Figura 71 ya se han explicado en el apartado 8.3.1, algunas de ellas tienen ligeras variaciones en la funcionalidad. La más importante es la clase DeviceScanFragment.kt, situada en el paquete scan. En dicha clase se concentra la funcionalidad de mostrar el dispositivo visible para que los pasajeros puedan encontrarlo, tanto de actualizar el nombre, matrícula y número de zonas del vehículo.

El diagrama de navegación de VehicleApp es el mostrado en la SEQ Figura * ARABIC:

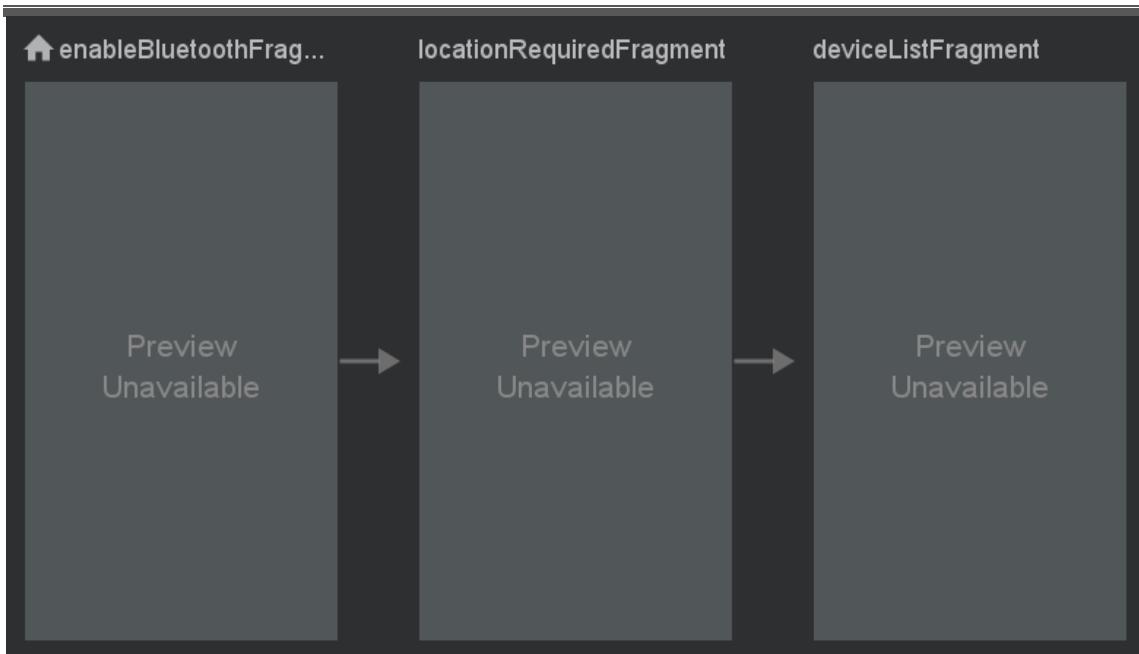


Figura 74 Diagrama de pantallas de VehicleApp.

Capítulo 9. Conclusiones y Ampliaciones

En este apartado se enumerarán las conclusiones a las que se ha llegado mediante la realización del trabajo, así como qué posibles ampliaciones se podrían realizar en un futuro.

9.1 Conclusiones

Se ha realizado una auditoría a las tarjetas de transporte público de Asturias, las cuales ya se sabía que eran inseguras, porque se habían clonado anteriormente, tal y como muestra la siguiente noticia (El Comercio, 2018), y se ha demostrado que son inseguras en base a pruebas empíricas, estudios, estándares y artículos.

Si bien la auditoría es la parte principal del Trabajo de Fin de Grado y en la que recayó la mayor parte del esfuerzo, se ha propuesto como alternativa al uso de tarjetas una aplicación que los pasajeros puedan llevar instalada en el móvil.

Hemos conseguido desarrollar una aplicación lo suficientemente compatible como para llegar a una cantidad de usuarios amplia y poder sustituir a las tarjetas vulnerables. La aplicación funciona con varios dispositivos al mismo tiempo sin errores, por lo tanto, se ha logrado lo que se esperaba: una prueba de concepto que demuestre que es posible una aplicación móvil para utilizar bonos de transporte público.

No obstante, la aplicación no está lista para ponerla en producción, pues aún tiene aspectos que deben ser desarrollados, mediante las mejoras que se proponen a continuación, en el apartado 9.2.

9.2 Ampliaciones

En este apartado se detallarán las posibles ampliaciones que se tienen en cuenta para el futuro del proyecto:

9.2.1 Desarrollo de la aplicación para iOS

Para llegar al máximo número de personas posible, es necesario desarrollar la aplicación para iOS. Esto supone un coste muy amplio pues supone aprender una tecnología totalmente nueva y se sale del alcance del proyecto. También se podría utilizar un framework como React Native o Flutter, que permiten realizar aplicaciones híbridas, pero esos frameworks no permiten trabajar con los módulos de bluetooth de los dispositivos de la misma manera que el lenguaje nativo Android o iOS.

Como la aplicación utiliza Bluetooth en lugar de NFC, se podría desarrollar su versión en iOS (para dispositivos iPhone) y funcionaría sin problemas de compatibilidad con los dispositivos Android de los vehículos.

9.2.2 Pagos reales por la pasarela de pago de PayPal

Se ha implementado el servicio de PayPal para permitir comprar los bonos de manera real, y así dar beneficios a la empresa de transporte que esté utilizando la aplicación (En este caso, la CTA).

No obstante, nuestra aplicación está en modo de desarrollo o “Sandbox”, tal y como indica nuestro centro de control de PayPal, en la Figura 75:

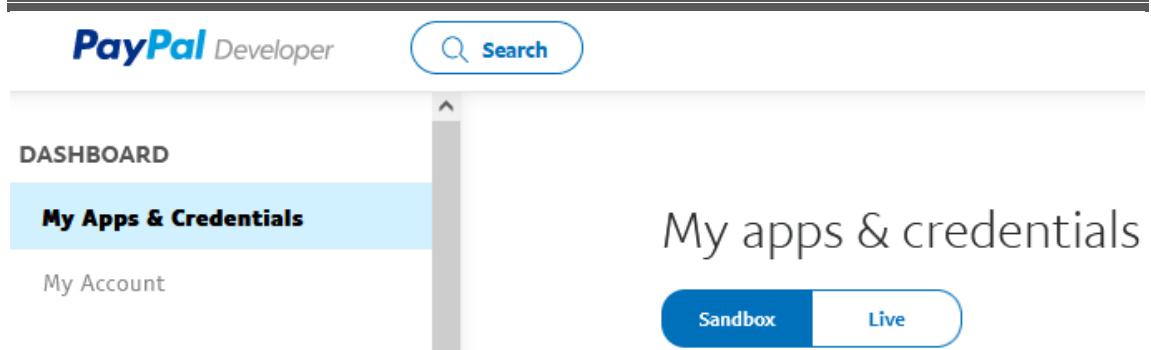


Figura 75 Centro de control de PayPal.

Lo cual significa que no llegará nada de dinero real a la cuenta de PayPal principal de la aplicación (la cuenta de vendedor), ya que se permite el pago con cuentas PayPal de desarrollador, con dinero ilimitado, como la mostrada en la Figura 76:

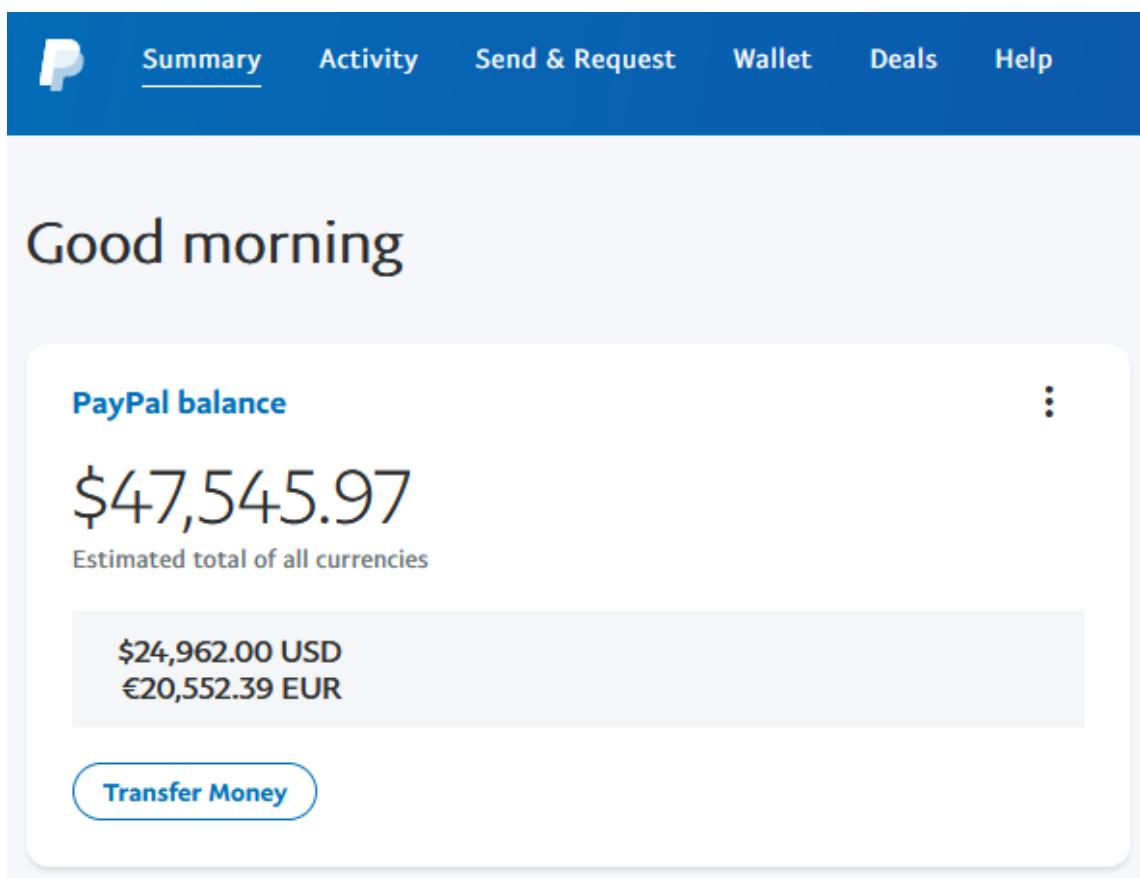


Figura 76 Cuenta de desarrollador de PayPal, con dinero ficticio, el cual solo se permite gastar en aplicaciones que estén en modo de desarrollador (Sandbox).

Por lo que, como posible ampliación, estaría transformar nuestra pasarela de pago en una pasarela en producción, con dinero real, lo cual habría que solicitar a PayPal para su aprobación.

9.2.3 Mejora del log de la aplicación

Actualmente la aplicación consta de un log en el que se va escribiendo cada vez que el usuario hace una operación. En un futuro estaría bien permitir al usuario enviarnos ese log en caso de que la aplicación diera algún error, para poder detectar posibles errores cuando la aplicación esté desplegada. De esa manera la solución de posibles errores se realizaría de manera más mantenible en un entorno de producción.

9.2.4 Realización de tests automáticos para el proyecto.

Actualmente, el proyecto es pequeño y todavía se puede probar a mano, porque tiene poca funcionalidad. No obstante, si en el futuro el desarrollo crece, se necesitarán pruebas automatizadas por los siguientes motivos:

- Al hacer cualquier cambio en la aplicación, podremos saber si ese cambio afecta a la funcionalidad anteriormente implementada sin necesidad de volver a probar la aplicación manualmente, simplemente tendremos que darle a un botón (o incluso podríamos tener una integración continua, como Travis (Contribuidores Travis, 2020) por ejemplo, en GitHub, que ejecuta los test automáticamente cada vez que se hace un commit), y los test nos dirán si los cambios han afectado negativamente a la funcionalidad anterior.
- Si la aplicación crece en funcionalidad, acaba siendo imposible probar la aplicación a mano, pues las posibilidades aumentan exponencialmente al aumentar el número de combinaciones entre casos de prueba posibles.

Para realizar los tests en Android, habría que hacer los tres tipos de test mostrados en la Figura 77:

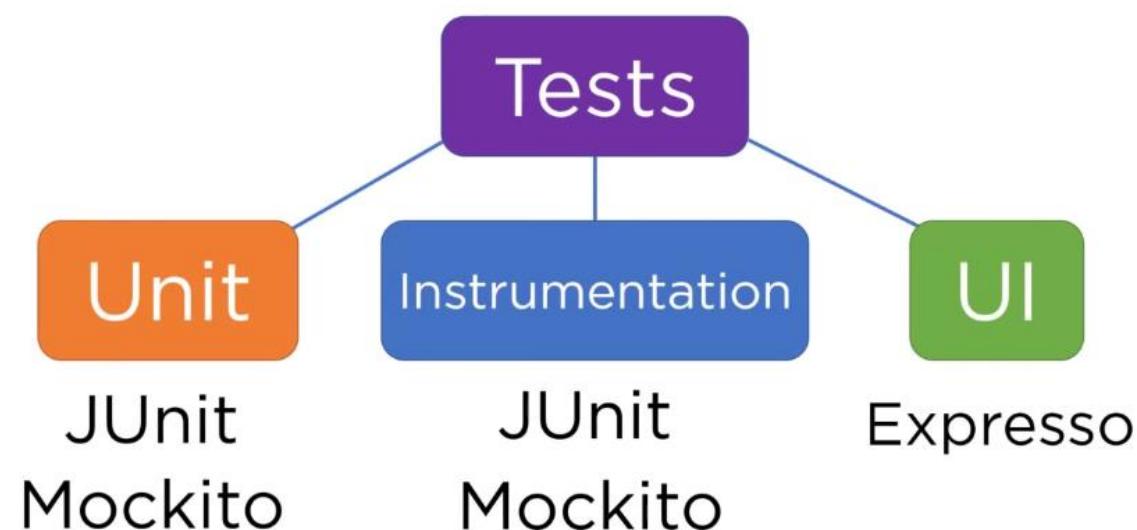


Figura 77 Principales tipos de test en Android.

Dichos tipos de pruebas se explican a continuación.

9.2.4.1 Unit test (o pruebas unitarias en español)

Se trata de tests que prueban la lógica de negocio de la aplicación, como el valor que devuelven ciertos métodos, por ejemplo, y están hechos en Java, tal y como muestra la Figura 78.



Local Computer
Java Virtual Machine (JVM)
Very fast

Figura 78 Resumen Unit tests.

Se ejecutan en local por lo que son rápidos. Se suele utilizar JUnit5. No necesitan de un emulador o dispositivo real para ejecutarse, pues se ejecutan en la máquina virtual de Java (JVM o Java Virtual Machine).

9.2.4.2 Instrumentation tests

Testean funcionalidad específica de Android, como actividades, fragmentos, contextos y servicios. Se suele utilizar JUnit4 ya que JUnit5 todavía no soporta este tipo de testing, como se muestra en la Figura 79.

Instrumentation

Similar to local unit tests

Need a real device or emulator

JUnit4, Mockito

Figura 79 Resumen Instrumentation tests.

Se requiere de una máquina virtual o un dispositivo Android real para poder ejecutar dichos tests.

9.2.4.3 UI test (o tests de interfaz de usuario)

Son test que simulan la interacción real con el usuario, pulsando botones, rellenando cuadros de texto, etc. Necesitan un emulador o un dispositivo Android real para poder ejecutarse, tal y como se muestra en la Figura 80.



Simulate a person using your app

Literally uses widgets

Real device or emulator

Expresso

Figura 80 Resumen UI tests.

Se suele utilizar la librería Expresso para realizar estos tests.

9.2.5 Auditar las tarjetas de otras provincias

Respecto a la auditoría, debido a que la misma se ha limitado a las tarjetas de la CTA, una posible ampliación sería comprobar la seguridad de las tarjetas de otras provincias de España, como Madrid, Valencia o Barcelona, e intentar clonarlas para determinar su seguridad.

Debido a que, aunque se hayan hecho pruebas con tarjetas de Madrid y Valencia, no se ha conseguido explotar vulnerabilidades en las mismas.

Capítulo 10. Planificación final del proyecto

El siguiente capítulo abarcará la planificación del desarrollo del proceso completo del Trabajo de Fin de Grado, incluyendo tanto la auditoría como las dos aplicaciones para Android realizadas.

Para una mejor organización y legibilidad tanto de los diagramas de Gantt como de este mismo apartado de este documento, se ha dividido la planificación en dos diagramas de Gantt, uno para la auditoría (desde el jueves 03/12/2020 hasta el martes 19/01/2021) y otro para las aplicaciones Android (desde el lunes 25/01/21 hasta el lunes 28/06/2021).

10.1 Planificación de la auditoría

La planificación de la auditoría se encuentra desglosada en detalle en el documento de Auditoría (Apartado 14. Planificación). No obstante, se muestra a continuación un resumen:

Desarrollo de la auditoría inicial:

- Fecha de inicio: jueves 03/12/2020.
- Fecha de fin: martes 19/01/2021.
- Duración en horas: 136 horas (34 días * 4 horas/día).

Desarrollo de la ampliación de la auditoría:

- Fecha de inicio: martes 24/02/2022.
- Fecha de fin: jueves 24/06/2022.
- Duración en horas: 376 horas (distribuidas irregularmente entre la fecha de inicio y la fecha de fin).

En total, la auditoría ha tenido una duración de **512 horas**.

10.2 Planificación de las aplicaciones Android

La planificación de las dos aplicaciones Android se encuentra disponible en el archivo “Documentación App/Archivos adjuntos/Planificación.mpp”.

En el archivo Planificación.mpp se encuentra el diagrama de Gantt con todas las tareas que se han llevado a cabo. En la Figura 81, se muestra una captura de pantalla de cómo se ve el archivo abierto con el Microsoft Project. Como podemos observar, el diagrama de Gantt se muestra a la derecha, representando gráficamente la duración de las tareas listadas a la izquierda:

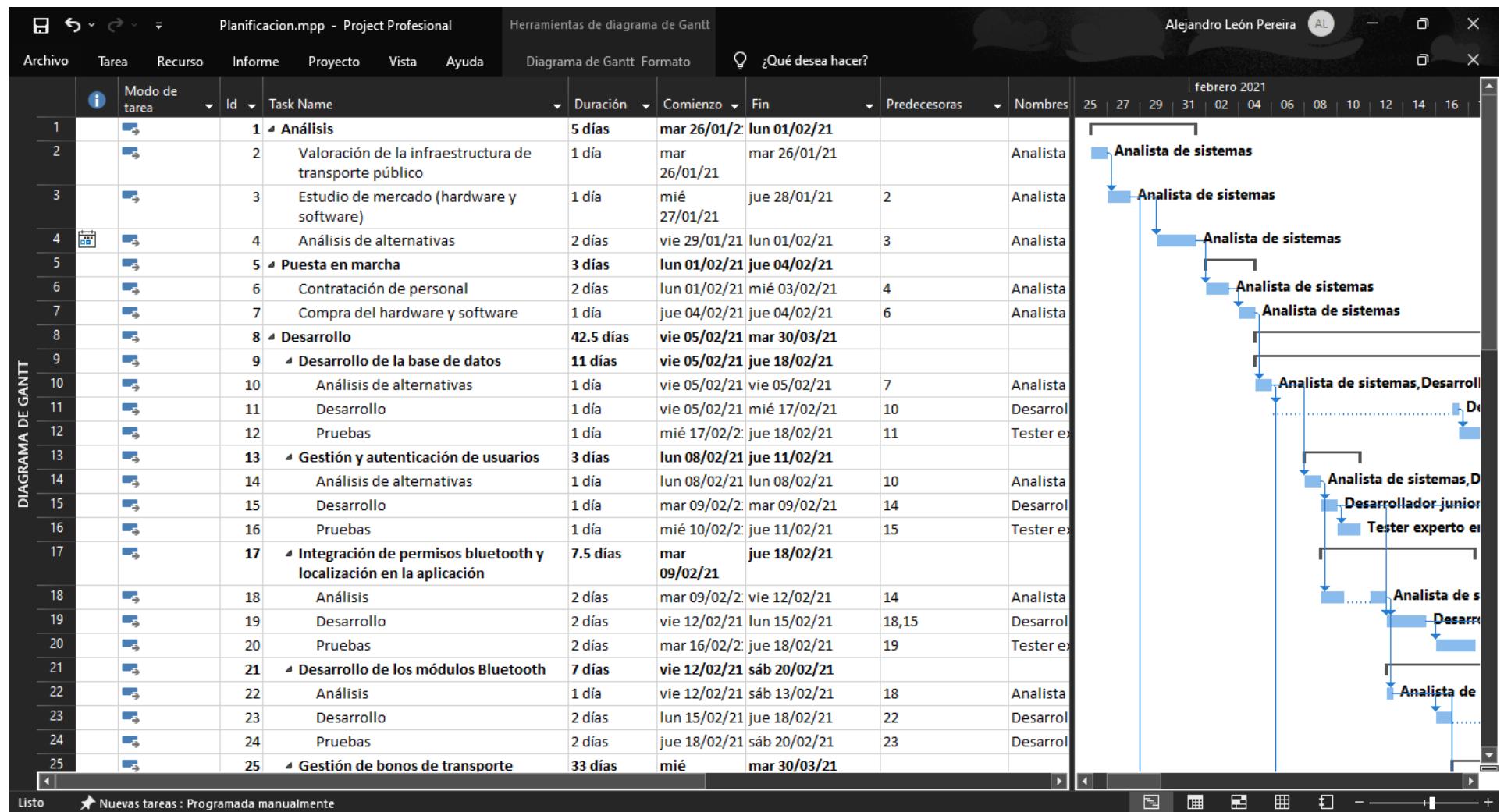


Figura 81 Archivo de planificación de las aplicaciones Android.

A continuación, se muestra en la Figura 82 y la Figura 83, la lista de todas las tareas comprendidas en la planificación de las aplicaciones Android.

Id	Task Name	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	▲ Análisis	5 días	mar 26/01/21	lun 01/02/21		
2	Valoración de la infraestructura de transporte público	1 día	mar 26/01/21	mar 26/01/21		Analista de sistemas junior
3	Estudio de mercado (hardware y software)	1 día	mié 27/01/21	jue 28/01/21	2	Analista de sistemas junior
4	Análisis de alternativas	2 días	vie 29/01/21	lun 01/02/21	3	Analista de sistemas junior
5	▲ Puesta en marcha	3 días	lun 01/02/21	jue 04/02/21		
6	Contratación de personal	2 días	lun 01/02/21	mié 03/02/21	4	Analista de sistemas junior
7	Compra del hardware y software	1 día	jue 04/02/21	jue 04/02/21	6	Analista de sistemas junior
8	▲ Desarrollo	42.5 días	vie 05/02/21	mar 30/03/21		
9	▲ Desarrollo de la base de datos	11 días	vie 05/02/21	jue 18/02/21		
10	Análisis de alternativas	1 día	vie 05/02/21	vie 05/02/21	7	Analista de sistemas junior, D
11	Desarrollo	1 día	vie 05/02/21	mié 17/02/21	10	Desarrollador junior android,
12	Pruebas	1 día	mié 17/02/21	jue 18/02/21	11	Tester junior de plataformas
13	▲ Gestión y autenticación de usuarios	3 días	lun 08/02/21	jue 11/02/21		
14	Análisis de alternativas	1 día	lun 08/02/21	lun 08/02/21	10	Analista de sistemas junior, D
15	Desarrollo	1 día	mar 09/02/21	mar 09/02/21	14	Desarrollador junior android
16	Pruebas	1 día	mié 10/02/21	jue 11/02/21	15	Tester junior de plataformas
17	▲ Integración de permisos bluetooth y localización en la aplicación	7.5 días	mar 09/02/21	jue 18/02/21		
18	Análisis	2 días	mar 09/02/21	vie 12/02/21	14	Analista de sistemas junior
19	Desarrollo	2 días	vie 12/02/21	lun 15/02/21	18,15	Desarrollador junior android
20	Pruebas	2 días	mar 16/02/21	jue 18/02/21	19	Tester junior de plataformas
21	▲ Desarrollo de los módulos Bluetooth	7 días	vie 12/02/21	sáb 20/02/21		
22	Análisis	1 día	vie 12/02/21	sáb 13/02/21	18	Analista de sistemas junior
23	Desarrollo	2 días	lun 15/02/21	jue 18/02/21	22	Desarrollador junior android
24	Pruebas	2 días	jue 18/02/21	sáb 20/02/21	23	Desarrollador junior android,
25	▲ Gestión de bonos de transporte público	33 días	mié 17/02/21	mar 30/03/21		
26	Análisis	1 día	mié 17/02/21	jue 18/02/21	22	Desarrollador junior android
27	Desarrollo	2 días	sáb 06/03/21	lun 08/03/21	26	Desarrollador junior android
28	Pruebas	2 días	sáb 27/03/21	mar 30/03/21	27	Tester junior de plataformas
29	▲ Conectividad entre PassengerApp y VehicleApp	22 días	vie 19/02/21	vie 19/03/21		
30	Análisis	1 día	vie 19/02/21	sáb 20/02/21	26	Analista de sistemas junior
31	Desarrollo	2 días	lun 15/03/21	mar 16/03/21	30,27	Desarrollador junior android
32	Pruebas	1 día	mié 17/03/21	vie 19/03/21	31	Tester junior de plataformas
33	▲ Integración de las aplicaciones Android con la base de datos	26.5 días	sáb 20/02/21	vie 26/03/21		
34	Análisis	1 día	sáb 20/02/21	mar 23/02/21	30	Analista de sistemas junior
35	Desarrollo	2 días	mar 23/03/21	jue 25/03/21	34,31	Desarrollador junior android
36	Pruebas	1 día	jue 25/03/21	vie 26/03/21	35	Tester junior de plataformas
37	Pruebas generales	2 días	vie 26/03/21	jue 01/04/21	15,19,23,12,31	Tester junior de plataformas
38	Despliegue en repositorio	1 día	vie 02/04/21	vie 02/04/21	37	Desarrollador junior android
39	Documentación del proyecto	1 día	vie 02/04/21	sáb 03/04/21	38	Analista de sistemas junior,D
40	Corrección de comentarios de los tutores	5 días	lun 13/06/22	sáb 18/06/22		

Figura 82 Tareas comprendidas en la planificación de las aplicaciones Android (Parte 1).

Planificación final del proyecto | Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas.

40	Corrección de comentarios de los tutores	5 días	lun 13/06/22	sáb 18/06/22			
41	Elaboración de presupuesto	2 días	sáb 18/06/22	mar 21/06/22	40		
42	Revisión final	2 días	mar 21/06/22	jue 23/06/22	41		
43	▪ Reuniones de seguimiento	403 días	jue 28/01/21	vie 24/06/22			
44	▪ Reunion de seguimiento 1	1 día	jue 28/01/21	vie 29/01/21			
45	Elaborar acta de reunion	1 día	jue 28/01/21	vie 29/01/21	3	Analista de sistemas junior,D	
46	▪ Reunion de seguimiento 2	1 día	lun 01/02/21	lun 01/02/21			
47	Elaborar acta de reunion	1 día	lun 01/02/21	lun 01/02/21		Analista de sistemas junior,D	
48	▪ Reunion de seguimiento 3	1 día	sáb 06/02/21	sáb 06/02/21			
49	Elaborar acta de reunion	1 día	sáb 06/02/21	sáb 06/02/21	10	Analista de sistemas junior,D	
50	▪ Reunion de seguimiento 4	1 día	lun 28/06/21	lun 28/06/21			
51	Elaborar acta de reunion	1 día	lun 15/02/21	lun 15/02/21		Analista de sistemas junior,D	
52	▪ Reunion de seguimiento 5	1 día	lun 22/02/21	lun 22/02/21			
53	Elaborar acta de reunion	1 día	lun 22/02/21	lun 22/02/21		Analista de sistemas junior,D	
54	▪ Reunion de seguimiento 6	1 día	lun 01/03/21	lun 01/03/21			
55	Elaborar acta de reunion	1 día	lun 01/03/21	lun 01/03/21		Analista de sistemas junior,D	
56	▪ Reunion de seguimiento 7	1 día	lun 08/03/21	lun 08/03/21			
57	Elaborar acta de reunion	1 día	lun 08/03/21	lun 08/03/21		Analista de sistemas junior,D	
58	▪ Reunion de seguimiento 8	1 día	lun 15/03/21	lun 15/03/21			
59	Elaborar acta de reunion	1 día	lun 15/03/21	lun 15/03/21		Analista de sistemas junior,D	
60	▪ Reunion de seguimiento 9	1 día	lun 22/03/21	lun 22/03/21			
61	Elaborar acta de reunion	1 día	lun 22/03/21	lun 22/03/21		Analista de sistemas junior,D	
62	▪ Reunion de seguimiento 10	1 día	lun 29/03/21	lun 29/03/21			
63	Elaborar acta de reunion	1 día	lun 29/03/21	lun 29/03/21		Analista de sistemas junior,D	
64	▪ Reunion de seguimiento 11	1 día	mar 06/04/21	mar 06/04/21			
65	Elaborar acta de reunion	1 día	mar 06/04/21	mar 06/04/21		Analista de sistemas junior,D	
66	▪ Reunion de seguimiento 12	1 día	lun 05/04/21	lun 05/04/21			
67	Elaborar acta de reunion	1 día	lun 05/04/21	lun 05/04/21	39	Analista de sistemas junior,D	
68	▪ Reunion de seguimiento 13 Final	2 días	jue 23/06/22	vie 24/06/22			
69	Elaborar acta de reunion	2 días	jue 23/06/22	vie 24/06/22			

Figura 83 Tareas comprendidas en la planificación de las aplicaciones Android (Parte 2).

En la pestaña “Hoja de recursos” (situada abajo a la derecha del programa Project), se encuentran los recursos humanos que se han estimado necesarios para la realización del proyecto, tal y como muestra la Figura 84:

Resource Name	Tipo	Etiqueta de . . .	Iniciales	Grupo	Max. Units	Std. Rate	Ovt. Rate	Cost/Use	Acumular	Calendario base
Desarrollador junior android	Trabajo		D		100%	20.00 €/hr	20.00 €/hr	0.00 €	Prorrateo	Standard
Tester junior de plataformas móviles	Trabajo		T		100%	25.00 €/hr	25.00 €/hr	0.00 €	Prorrateo	Standard
Desarrollador junior de bases de datos	Trabajo		BD		100%	20.00 €/hr	20.00 €/hr	0.00 €	Prorrateo	Standard
Analista de sistemas junior	Trabajo		A		100%	25.00 €/hr	25.00 €/hr	0.00 €	Prorrateo	Standard

Figura 84 División en roles de las tareas de la planificación.

Para el desarrollo de la planificación, he dividido el trabajo en cuatro roles:

- Desarrollador Junior Android: Será el desarrollador encargado de codificar la mayor parte del código de las aplicaciones Android.
- Tester junior de plataformas móviles: Será el encargado de garantizar el correcto funcionamiento y comportamiento de todos los ámbitos del desarrollo de las aplicaciones Android. Su función no solo será realizar test unitarios, sino depurar la aplicación en busca de anomalías, arreglar las mismas, y blindar el sistema contra errores.
- Desarrollador junior de Bases de Datos: Será el encargado no solo de analizar cuál de las alternativas disponibles en el mercado es adecuada para el proyecto sino además de desarrollar la base de datos del mismo.
- Analista de sistemas junior: Persona que llevará las riendas del proyecto entorno a qué tecnologías utilizar para cada determinada tarea.

Esto es debido a que en el desarrollo de aplicaciones las tareas a realizar son muy diferentes entre sí y he creído conveniente asignar cada tarea a uno o varios roles en función de su naturaleza. Los precios por hora provienen de una empresa real en la que he trabajado.

Como se puede ver en la Figura 84, cada rol tiene un precio distinto por hora. Por ejemplo, el desarrollador junior Android tiene un precio por hora de 20.00€, mientras que el tester junior de plataformas móviles tiene un precio por hora de 25.00€. Esto afectará a la estimación de costes de la planificación, como se muestra en la Figura 85:

Estadísticas del proyecto 'Planificacion.mpp'			
	Comienzo	Fin	
Actual	mar 26/01/21	vie 24/06/22	
Previsto	NOD	NOD	
Real	NOD	NOD	
Variación	0d	0d	
	Duración	Trabajo	Costo
Actual	405d	380h	8,620.00 €
Previsto	0d	0h	0.00 €
Real	0d	0h	0.00 €
Restante	405d	380h	8,620.00 €

Porcentaje completado:
Duración: 0% Trabajo: 0%

Cerrar

Figura 85 Resumen de la planificación de las aplicaciones Android.

Como podemos ver en la Figura 85, la planificación ha tenido una duración total de 405 días, o 380 horas (a una media de 1 hora diaria aproximadamente). El coste total del desarrollo del proyecto son 8,620.00€.

La planificación del proyecto se comprende entre las fechas 26/01/2021 y 24/06/2022.

Capítulo 11. Presupuesto

En este apartado se tratará tanto el presupuesto de la auditoría, como el presupuesto de las aplicaciones para Android. El presupuesto total será la suma de los dos presupuestos.

11.1 Presupuesto de la auditoría

Cabe aclarar que el presupuesto se ha realizado en base a lo que se le cobraría a una empresa o entidad que contratara el servicio de la auditoría. Como este proyecto ha sido realizado para un trabajo de fin de grado sin ánimo de lucro, en realidad no se ha cobrado nada a nadie. No obstante, se adjunta a continuación el presupuesto que habría que facturarle a una empresa contratante.

El presupuesto final es **5.144,74€** los cuales se desglosan en los siguientes gastos:

11.1.1 Gastos materiales

Se ha necesitado el siguiente material:

- Lector ACR122u (Amazon, 2022): 44,95€.
- 7 tarjetas de transporte público: 10,50€ (1,50€ cada una).
- 1 recarga de 10 viajes de 1 zonas sobre una de las tarjetas: 9,00€.
- USB de 16gb para usar como alojamiento del sistema operativo Linux: 10,00€.
- Portátil Lenovo M330 (precio de compra de 1.079,10€): Valor del portátil por el tiempo utilizado (depreciación): 69,56€.
- Gasto energético del portátil utilizado: 53,76€.
- Licencia Windows 10 Professional 64 bits (Microsoft, Página web principal de Windows 10, 2022): 14,90€.
- Los programas de Microsoft Office van incluidos dentro de las prestaciones de la Universidad de Oviedo.

En total, los gastos suman $44,95\text{€} + 10,50\text{€} + 9,00\text{€} + 10,00\text{€} + 69,56\text{€} + 53,76\text{€} + 14,90\text{€} = \underline{\underline{212,67\text{€}}}$.

La amortización del ordenador portátil utilizado en la auditoria se ha realizado de la siguiente manera. Teniendo en cuenta:

- El coste del ordenador: 1.079,10€.
- La duración del proceso de la auditoria ha sido en total 167 días, distribuidos en los siguientes intervalos de tiempo:
 - o Desde el jueves 03/12/2020 hasta el martes 19/01/2021 (47 días).
 - o Desde el jueves 24/02/2022 hasta el viernes 24/06/2022 (120 días).

No obstante, no sería justo hacer los cálculos con los 167 días totales que ha durado la auditoría, pues cada día no ha tenido una duración de 8 horas (jornada estándar de trabajo), sino que han tenido duraciones reducidas (en torno a 4 horas diarias aproximadamente). Por lo tanto, para hacer los cálculos, se han convertido las 512

horas totales trabajadas en la auditoría (tal y como explica el apartado 10.1) días laborables de 8 horas de trabajo.

512 horas equivalen, por tanto, a 64 días de trabajo. Tomando sábados y domingos como días no laborables, en un mes se trabajan aproximadamente 20 días (4 semanas * 5 días). Por tanto, 64 días laborables equivalen a 3 meses y 4 días (94 días contando sábados y domingos).

- La vida útil del ordenador con respecto a su uso para fines relacionados con la auditoría es de aproximadamente 4 años. Dividiendo el coste del ordenador (1.079,10€) entre 1.460 días (los días que hay en 4 años) nos queda una media de **0,74€/día**.
- Teniendo en cuenta que un ordenador promedio, en 8 horas gasta 2,2kWh, y la media en España a 23 de diciembre de 2021 es de 0,37843 €/kWh, nos queda que, el ordenador portátil utilizado gasta, de luz, aproximadamente **0,84€/día**.
- El gasto total de luz por parte del ordenador serían $0,84\text{€}/\text{día} * 64 \text{ días} = \textbf{\underline{53,76\text{€}}}$.
- El gasto por el valor del portátil (depreciación) sería $0,74\text{€}/\text{día} * 94 \text{ días} (3 \text{ meses y } 4 \text{ días}) = \textbf{\underline{69,56\text{€}}}$.

Es decir, el total serían 53,76€ (gasto energético) + 69,56€ (depreciación del ordenador) = **123,32€**.

11.1.2 Gastos de recursos humanos

El número de horas que yo, Alejandro León Pereira, autor de esta auditoría, he dedicado al proyecto, son 512 horas (tal y como se muestra en el apartado [11. Planificación](#) de este documento).

El precio por hora del trabajo como auditor que he seleccionado son 25,00 €/hora, el cual es un precio estándar para trabajadores dedicados al sector de la informática con perfil junior/principiante. Si bien los precios promedio de diferentes empresas de auditoría rondan los 50,00 €/hora, se ha tenido en cuenta que soy un auditor principiante (o junior) en la materia, por lo que no podría cobrar la misma cantidad que una empresa especializada en la materia de la auditoría, ya que he empleado la mayor parte del tiempo en labores de aprendizaje, mientras que un auditor profesional aprovecharía mucho más el tiempo.

Por lo tanto, el coste total en recursos humanos sería el siguiente:

$$512 \text{ horas} * 25,00 \text{ €/hora} = \textbf{\underline{12.800,00 €}}$$

11.1.3 Gastos totales del desarrollo de la auditoría

La suma de los gastos de recursos humanos más los gastos materiales sería la siguiente:

$$12.800,00 \text{ € (gastos de recursos humanos)} + 212,67 \text{ € (gastos materiales)} = \textbf{\underline{13.012,67 \text{ €}}}$$

Respecto al beneficio industrial, si bien he desarrollado el proyecto por mi cuenta, aplicaré un beneficio industrial, el cual es el que se llevaría la empresa de auditores, en el caso de que yo

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Presupuesto

estuviera trabajando para alguna. El beneficio industrial elegido ha sido del 7%, y por lo tanto equivale a 910,89€ (13.012,63€ * 0,07€).

El presupuesto total de la auditoría incluyendo el beneficio industrial sería **13.923,56€** (13.012,67 + 910,89€).

El IVA sobre el presupuesto total sería un 21% de 13.923,56€. Es decir, 2.923,95€.

El presupuesto total de la auditoría contando el IVA son **16.847,51€**.

El desglose resumido se encuentra en el archivo “Documentación App/Archivos adjuntos/Presupuesto.xlsx”, y se puede observar en la Figura 86:

RESUMEN DEL PRESUPUESTO: AUDITORÍA	
Costes Materiales (C)	212,67 €
Coste Recursos Humanos	12.800,00 €
Beneficio Industrial (7%)	910,89 €
SUBTOTAL=	13.923,56 €
IVA 21%	2.923,95 €
TOTAL=	16.847,50 €

Figura 86 Desglose del presupuesto de la auditoría.

11.2 Presupuesto de las aplicaciones Android

En este apartado se recoge el presupuesto de las aplicaciones para Android desarrolladas.

11.2.1 Archivos referenciados

Hoja Excel sobre los presupuestos: Documentación app/Archivos adjuntos/Presupuesto.xlsx
Documentación del proyecto al que está asociado: Documentación app/Documentación TFG.docx

11.2.2 Gastos materiales

Para realizar el desarrollo del software para Android se ha requerido del siguiente material:

1. Depreciación del portátil Lenovo: **55,50€**.
2. Gasto energético del portátil Lenovo: **40,32€**.
3. Para desarrollar las dos aplicaciones móviles y testearlas en dos dispositivos android reales se ha necesitado adquirir dos dispositivos android. Los teléfonos adquiridos son de la marca Xiaomi, modelo Redmi 9, y se han adquirido por 122,00€ cada uno. Actualmente, en el mercado de segunda mano se pueden encontrar en buen estado por 150€, ya que el precio de los mismos nuevos se ha elevado hasta los 200€. Por lo tanto, no se ha contabilizado en el presupuesto el valor de los teléfonos móviles.

No se han añadido gastos por el consumo energético de los teléfonos, ya que estos van enchufados por USB al ordenador, y por lo tanto van incluidos en el consumo energético del ordenador portátil Lenovo.

4. Para desarrollar el software no se han utilizado licencias de pago (más allá de Windows) ya que todo el software utilizado es gratuito (Java, Android Studio, Librerías de Google, etc.). En este caso y al ser un proyecto en conjunto (Ya que el TFG incluye tanto la auditoría como las aplicaciones), como la licencia de Windows 10 ya se incluyó en el presupuesto de auditoría, no se volverá a incluir aquí debido a que el proyecto en su conjunto va dirigido a la misma entidad, y no tendría sentido cobrar dos veces la licencia de Windows. Los programas de Microsoft Office van incluidos dentro de las prestaciones de la Universidad de Oviedo.

Por lo tanto, se concluye que el total de gastos materiales para el desarrollo de las aplicaciones para Android es **95,82€**.

La amortización del portátil se ha realizado igual que en el presupuesto de la auditoría, en el apartado 11.1.1, teniendo en cuenta:

- El coste del ordenador: 1070,10€.
- Total de horas empleadas en el desarrollo de las aplicaciones: **380 horas**.
- 380 horas de trabajo equivalen, aproximadamente, a **48 días laborables** (días de 8 horas de trabajo).
- Tomando como días laborables de una semana 5, 48 días de trabajo corresponden, aproximadamente, a 10 semanas laborables. Es decir, **2 meses y medio**, aproximadamente.
- En el apartado 11.1.1 se ha calculado que el gasto de depreciación del portátil son 0,74€/día. Por lo tanto, el gasto de depreciación por 2 meses y medio son aproximadamente $0,74\text{€} * 75 \text{ días} = \textbf{\underline{55,50€}}$.
- En el apartado 11.1.1 se ha calculado que, al día, el portátil gasta aproximadamente 0,84€. Por lo tanto, el gasto durante 48 días en los que se ha trabajado durante 8 horas, sería $0,84\text{€} * 48 = \textbf{\underline{40,32€}}$.

11.2.3 Gastos de recursos humanos

El presupuesto final (para los gastos de recursos humanos, sin contar los gastos materiales) es de 8.620,00€ tal y como se muestra en el fichero Presupuesto.xlsx de la Figura 87, así como en el apartado 10.2.

	Precio unitario				
	Desarrollador junior Android	Desarrollador junior de bases de datos	Tester junior de plataformas móviles	Analista de sistemas junior	
	20,00 €	20,00 €	25,00 €	25,00 €	
Número de horas					
Concepto	Desarrollador junior Android	Desarrollador junior de bases de datos	Tester junior de plataformas móviles	Analista de sistemas junior	Coste Total Concepto
Análisis y estudio de alternativas	14	17	19	11	1.370,00 €
Diseño de la Base de Datos	12	10	18	16	1.290,00 €
Implementación de PassengerApp	17	13	23	13	1.500,00 €
Implementación de VehicleApp	21	15	17	14	1.495,00 €
Testing	22	20	37	2	1.815,00 €
Desarrollo de documentación	10	15	13	13	1.150,00 €
TOTAL HORAS (de cada rol)	96	90	127	69	8.620,00 €
TOTAL HORAS (todos los roles)	382				Subtotal

Figura 87 Presupuesto final del desarrollo de las dos aplicaciones Android.

Cabe destacar que, si bien existen varios roles en el desarrollo del proyecto, yo, como autor del mismo, he interpretado todos y cada uno de ellos para la realización del mismo. Los precios por hora, como se ha especificado en el apartado 10.2, provienen de una empresa real en la que he trabajado.

11.2.4 Gastos totales del desarrollo de las aplicaciones Android

La suma de los gastos humanos y materiales del desarrollo de las aplicaciones sería la siguiente:

$$8.620,00\text{€} \text{ (gastos de recursos humanos)} + 95,82\text{€} \text{ (gastos materiales)} = \underline{\underline{8.715,82\text{€}}}.$$

El beneficio industrial elegido ha sido del 7%. El 7% de 8.715,82€ es 610,11€. Por lo tanto, el coste incluyendo el beneficio industrial sería de 9.325,93€.

El 21% de IVA sobre la cifra anterior es $9.325,93\text{€} * 0,21 = 1.958,45\text{€}$.

El coste total del desarrollo de las aplicaciones Android, tal y como muestra la Figura 88, procedente del archivo “Documentación App/Archivos adjuntos/Presupuesto.xlsx”, es 11.284,38€.

RESUMEN DEL PRESUPUESTO: APLICACIONES ANDROID		
Costes Materiales (C)		95,82 €
Coste Recursos Humanos (D)		8.620,00 €
Beneficio Industrial (7%)		610,11 €
SUBTOTAL=		9.325,93 €
IVA 21%		1.958,44 €
TOTAL=		11.284,37 €

Figura 88 Resumen del presupuesto de las aplicaciones Android.

11.3 Presupuesto total

En este capítulo se calculará el presupuesto total a pagar por la empresa contratante.

11.3.1 Suma de los presupuestos de la auditoría y las aplicaciones Android

El presupuesto resultante incluyendo tanto la auditoría, como las aplicaciones móviles, como el IVA y el beneficio industrial, es el siguiente:

16.847,51€ (coste total auditoría) + 11.284,38€ (coste total aplicaciones) = 28.131,89€.

11.3.2 Gastos anuales a pagar por los servicios en la nube

Es necesario considerar los gastos que acarrearía tener la aplicación en producción durante los años. Para ello, se ha calculado una aproximación de lo que tendría que pagar anualmente la CTA, basándose en el número de usuarios aproximado que utiliza el transporte público en Asturias.

Para poder poner la aplicación en producción, hacen falta tomar dos consideraciones en cuenta:

- La aplicación del vehículo. Si bien en un futuro se podría comprar un dispositivo Android para cada vehículo, para reducir los costes, vamos a suponer que inicialmente el conductor del vehículo utilizará su propio terminal como aplicación del vehículo.
- Los gastos propios de la base de datos de Google Firebase, los cuales serán calculados a continuación.

En la siguiente referencia (SOCIEDAD ASTURIANA DE ESTUDIOS ECONÓMICOS E INDUSTRIALES, 2022) podemos obtener las cantidades de viajes llevados a cabo en Asturias durante los últimos años, tal y como muestra la Figura 89:

	TOTAL	Transporte ferroviario	Autobús interurbano	Autobús urbano
Total anual				
2,015	18,274,444	2,694,733	8,952,395	6,627,316
2,016	18,087,857	2,657,344	8,751,455	6,679,058
2,017	18,244,031	2,668,438	8,644,427	6,931,166
2,018	18,622,039	2,604,915	8,724,467	7,292,657
2,019	18,995,429	2,627,518	8,837,030	7,530,881
2,020	11,104,627	1,777,578	4,848,272	4,478,777
2,021	13,386,042	2,044,142	5,791,907	5,549,993

Figura 89 Cantidad de viajes llevados a cabo en Asturias durante los últimos años.

La media de los totales de los últimos años es 16.673.496 viajes por año, contando los tres tipos de transporte. No obstante, debido a la pandemia “COVID-19” (Organización Mundial de la Salud, 2022), los viajes de 2020 y 2021 han decrementado notablemente respecto a 2019. Por lo tanto, para ponernos en el peor de los casos en el presupuesto, se harán los cálculos con la cifra mayor, es decir, la de 2019, es decir, 18.995.429 viajes.

Con nuestro sistema de aplicaciones Android, cada viaje supone leer (1 vez) y escribir (1 vez) en base de datos. Además, se tendrá en cuenta que los viajes se comprarán de 10 en 10, y por tanto por cada 10 viajes se producirá 1 lectura más y otra escritura más.

Por lo tanto, 18.995.429 viajes equivalen a las siguientes lecturas y escrituras:

En relación con las veces en las que un usuario se suba a un determinado vehículo de transporte público, se efectuarán:

- 18.995.429 lecturas (para saber cuántos viajes tiene el pasajero).
- 18.995.429 escrituras (para decrementar el número de viajes del pasajero).

En relación con las veces en las que un usuario comprará bonos de transporte público:

- 1.899.543 lecturas (resultado de dividir 18.995.429 entre 10 y redondear). Estas escrituras se realizan para saber cuántos viajes tiene el pasajero al momento de comprar un bono de 10 viajes, es decir, el mínimo existente, y, por tanto, el caso en el que más escrituras se efectúan. Es decir, poniéndonos en el caso de que los bonos se compren de 10 en 10, se efectuarían 1.899.543 lecturas.
- 1.899.543 escrituras (para aumentar en 10 unidades el número de viajes de un pasajero al comprar un bono de 10 viajes).

En total, son 20.894.972 lecturas y 20.894.972 escrituras.

Google Firebase permite hacer 50.000 lecturas y 20.000 escrituras de manera gratuita cada mes, tal y como indican las políticas de Google Firebase en la siguiente referencia (Google, Consulta un ejemplo de precios de Cloud Firestore, 2022). Por lo tanto, al año, Google Firebase permite un total de 600.000 lecturas y 240.000 escrituras/año.

Esto quiere decir, que se tendrán que pagar las siguientes operaciones:

- $20.894.972$ (lecturas anuales aproximadas de la CTA) - 600.000 (lecturas gratuitas) = $20.294.972$ (lecturas a pagar cada año aproximadamente).
- $20.894.972$ (escrituras anuales aproximadas de la CTA) - 24.000 (escrituras gratuitas) = $20.870.972$ (escrituras a pagar cada año aproximadamente).

Google Firebase tiene un precio de $0,057\text{€}$ (o $0,06\$$) por cada 100.000 lecturas, y $0,17\text{€}$ (o $0,18\$$) por cada 100.000 escrituras, tal y como muestra la siguiente referencia (Google, Consulta un ejemplo de precios de Cloud Firestore, 2022). Por lo tanto, nos quedarán a pagar las siguientes cifras:

- $20.294.972$ lecturas * $0,06\text{€} / 100.000$ lecturas = $12,18\text{€}$.
- $20.870.972$ escrituras * $0,17\text{€} / 100.000$ escrituras = $35,48\text{€}$.

En total, en un año la CTA tendrá que pagar aproximadamente $47,66\text{€}$ por las lecturas y escrituras en la base de datos del proyecto.

Respecto al almacenamiento de los propios datos de los usuarios registrados, Google Firebase también aplica un impuesto, que se calculará a continuación.

En el siguiente artículo (RTPA, 2022), se asegura que el número de usuarios de la CTA a 10 de marzo de 2022, es de 37.405 personas. En las políticas de Google Firebase se especifica que una app instalada en 50.000 dispositivos pagaría al mes un coste de almacenamiento de 1€ ($1,04\$$). Es decir, al año serían 12€ . Por lo tanto, con aproximadamente 37.405 usuarios se pagaría al año $8,98\text{€}$ por tener a los usuarios registrados con su email/contraseña o cuenta de Google, además de sus viajes almacenados.

Por lo tanto, contando todos los costes, la CTA tendría que pagar al año $47,66\text{€} + 8,98\text{€} = 56,64\text{€}$ aproximadamente.

A esos $56,64\text{€}$ habría que sumarles el IVA, el cual sería el 21% de $56,64\text{€}$, es decir, $11,90\text{€}$.

Por lo tanto, el coste total a pagar por los servicios de Google Firebase, contando impuestos, sería de $68,54\text{€}$.

11.3.3 Presupuesto total final

El presupuesto total serían los **$28.131,89\text{€}$** de la auditoría y las aplicaciones móviles, incluyendo impuestos, y calculado en el punto 11.3.1.

El pago de dicha cantidad se efectuará una única vez, sin embargo, para poder mantener la aplicación en producción, será necesario abonar anualmente $68,54\text{€}$ (calculados en el apartado 11.3.2) para pagar los servicios en la nube de Google Firebase, teniendo en cuenta el volumen de usuarios de la CTA.

Capítulo 12. Referencias Bibliográficas

La siguiente bibliografía está activa y accesible a fecha de 15 de noviembre de 2021.

Adrián Raya. (2022, 09 17). *Prácticamente todo lo que tenga Bluetooth es vulnerable a este nuevo ataque.* Retrieved from elespanol.com: https://www.elespanol.com/omicrono/software/20200917/practicamente-bluetooth-vulnerable-nuevo-ataque/521448782_0.html

Amazon. (2022). *Amazon NFC ACR122U USB RFID.* Retrieved from amazon.es: https://www.amazon.es/dp/B07MVBHR29/ref=cm_sw_em_r_mt_dp_6WYKZ9TP3SZXYD5CWTJT?_encoding=UTF8&psc=1

Consorcio de Transportes de Madrid. (2022). *Sitio web del Consorcio de Transportes de Madrid.* Retrieved from crt.m.es: <https://www.crtm.es/?url=https://www.crtm.es/>

Contribuidores de nfc-mfclassic. (2020, 05 30). *Comando nfc-mfclassic en GitHub.* Retrieved from github.com: <https://github.com/nfc-tools/libnfc/blob/master/utils/nfc-mfclassic.c>

Contribuidores libnfc. (2021, 09 20). *Repositorio de Github de la librería libnfc.* Retrieved from github.com: <https://github.com/nfc-tools/libnfc>

Contribuidores nfc-tools. (2022, 05 21). *Repositorio de Github de la librería nfc-tools.* Retrieved from github.com: <https://github.com/nfc-tools>

Contribuidores scrcpy. (2022, 06 20). *Repositorio de Github del programa scrcpy.* Retrieved from github.com: <https://github.com/Genymobile/scrcpy>

Contribuidores Travis. (2020, 05 17). *Sistema de integración continua Travis.* Retrieved from github.com: <https://github.com/travis-ci/travis-ci>

Contribuidores Wikipedia. (2022, 03 08). *Artículo de la Wikipedia de Android Beam.* Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Android_Beam

Contribuidores Wikipedia. (2022, 06 09). *Artículo de la Wikipedia de Bluetooth.* Retrieved from es.wikipedia.org: <https://es.wikipedia.org/wiki/Bluetooth>

Contribuidores Wikipedia. (2022, 05 15). *Página de la Wikipedia de Android Marshmallow.* Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Android_Marshmallow

CTA. (2022). *Mapa de zonas de la CTA.* Retrieved from consorcioasturias.com: <https://www.consocrioasturias.com/es/portal.do?NM=2&IDM=119>

- El Comercio. (2018, Abril 29). *El Comercio*. Retrieved from <https://www.elcomercio.es/asturias/detenidos-seis-jovenes-falsificar-tarjeta-consorcio-transportes-asturias-20180429002300-ntvo.html>
- Google. (2021, 03 30). *Calidad básica de las apps*. Retrieved from developer.android.com: <https://developer.android.com/docs/quality-guidelines/core-app-quality?hl=es-419>
- Google. (2021, 05 03). *Cómo ejecutar tu app*. Retrieved from developer.android.com: <https://developer.android.com/training/basics/firstapp/running-app>
- Google. (2022, 02 21). *Cómo instalar controladores USB de OEM*. Retrieved from developer.android.com: <https://developer.android.com/studio/run/oem-usb>
- Google. (2022, 02 21). *Cómo instalar controladores USB de OEM*. Retrieved from developer.android.com: <https://developer.android.com/studio/run/oem-usb>
- Google. (2022, 01 26). *Consulta un ejemplo de precios de Cloud Firestore*. Retrieved from firebase.google.com: <https://firebase.google.com/docs/firestore/billing-example?hl=es-419#medium-1m-installs>
- Google. (2022, 06 21). *Documentación de Firestore*. Retrieved from firebase.google.com: <https://firebase.google.com/docs/firestore>
- Google. (2022, 03 30). *Integración de login con Google en aplicaciones*. Retrieved from developers.google.com: <https://developers.google.com/identity/sign-in/web/sign-in>
- Google. (2022). *Sitio web principal de Android Studio*. Retrieved from developer.android.com: <https://developer.android.com/studio>
- Google. (2022, 05 20). *Transfiere datos entre proyectos*. Retrieved from firebase.google.com: <https://firebase.google.com/docs/firestore/manage-data/move-data>
- Grupo TMB. (2022). *TMB Barcelona*. Retrieved from [tmb.cat: https://www.tmb.cat/es/home](https://www.tmb.cat/es/home)
- Jeremy O'Donoghue. (2021). *¿Funciona el NFC de Apple con el de Android?* Retrieved from quora.com: <https://www.quora.com/Does-Apple-have-a-NFC-beam-like-on-Android-devices-And-if-so-does-NFC-work-with-other-Android-phones>
- Jet Brains. (2022). *Lenguaje de programación Kotlin*. Retrieved from kotlinlang.org/
- Joe Maring, J. J. (2022, 01 04). *Como usar nearby share en tu dispositivo Android*. Retrieved from androidcentral.com: <https://www.androidcentral.com/how-use-nearby-share-your-android-phone>
- Lea Verou. (2022). *Calculadora de ratio de contraste*. Retrieved from contrast-ratio.com: <https://contrast-ratio.com/>
- León Pereira, A. (2022, 06 21). *Repositorio de Github de PassengerApp y VehicleApp*. Retrieved from github.com: <https://github.com/alexlo/tfg>

Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Referencias Bibliográficas

Microsoft. (2022). *Página web principal de Windows 10*. Retrieved from microsoft.com: <https://www.microsoft.com/es-es/software-download/windows10>

Microsoft. (2022). *Sistema operativo Windows 10*. Retrieved from microsoft.com: <https://www.microsoft.com/es-es/software-download/windows10>

NXP Semiconductors. (2022). *Estandar del fabricante de Mifare Classicx*. Retrieved from nxp.com: https://www.nxp.com/products/rfid-nfc/mifare-hf/mifare-classic:MC_41863

Oracle. (1997, 09). *Java Code Conventions*. Retrieved from oracle.com: <https://www.oracle.com/technetwork/java/codeconventions-150003.pdf>

Oracle. (2022). *Página principal de Java*. Retrieved from java.com: <https://www.java.com/es/>

Oracle. (2022). *Sitio web de Java*. Retrieved from java.com: <https://www.java.com/es/>

Organización Mundial de la Salud. (2022). *Brote de enfermedad por coronavirus (COVID-19)*. Retrieved from who.int: <https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019>

Roche, E. (2000, 08). *Explicación del lenguaje XML*. Retrieved from hbr.org: <https://hbr.org/2000/07/explaining-xml>

RTPA. (2022, 03 19). *Aumenta el número de usuarios del transporte público por los combustibles*. Retrieved from rtpa.es: https://www.rtpa.es/noticias-asturias:Aumenta-el-numero-de-usuarios-del-transporte-publico-por-los-combustibles_111647696900.html

S.A.U., E. M. (2022). *Sitio web de EMT Valencia*. Retrieved from emtvalencia.es: <https://www.emtvalencia.es/ciudadano/index.php>

SOCIEDAD ASTURIANA DE ESTUDIOS ECONÓMICOS E INDUSTRIALES. (2022, 01 20). *Número mensual del viajeros del Consorcio de Transportes de Asturias (CTA) según medio de transporte*. Retrieved from sadei.es: https://www.sadei.es/numero-de-viajeros-del-cta/noticias/numero-de-viajeros-del-cta_174_21_202_0_1_in.html

Square Inc. (2022). *Seguridad de la tecnología NFC*. Retrieved from nearfieldcommunication.org: <http://nearfieldcommunication.org/nfc-security.html>

Torvalds, L. (2022). *Sistema de control de versiones Git*. Retrieved from git-scm.com: <https://git-scm.com/>

Yusef Hassan Montero, F. M. (2003, 03 30). *Guía de Evaluación Heurística de Sitios Web*. Retrieved from nosolousabilidad.com: <https://www.nosolousabilidad.com/articulos/heuristica.htm>

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

A continuación, se muestran todos los términos que se consideran importantes en la aplicación con una descripción breve de su significado.

- **Log:** Es un archivo donde se deja constancia cada vez que el usuario realiza una operación. El objetivo es saber qué se hizo y cómo se hizo para poder averiguar más fácilmente la causa de posibles errores.
- **NFC:** Del inglés Near Field Communication, es un estándar que permite el intercambio de información entre dispositivos en una distancia corta (menor a 5 cm).
- **Test:** Prueba. Se utilizan pruebas para comprobar que el software hace lo que tiene que hacer.
- **Tester:** Persona encargada de desarrollar pruebas para un determinado software.
- **Senior:** Programador de software experimentado.
- **Junior:** Programador de software principiante.

13.2 Contenido Entregado

13.2.1 Contenidos

El contenido entregado se divide en dos carpetas: Documentación Auditoría y Documentación App, tal y como se muestra en la Figura 90:

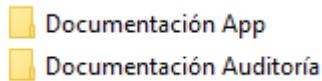


Figura 90 Carpetas principales del proyecto.

A continuación, se detallará el contenido de cada una de las carpetas de la Figura 90.

13.2.1.1 Documentación App

Dentro de Documentación App se encuentra este documento, junto con una carpeta de archivos adjuntos mostrada en la Figura 91:

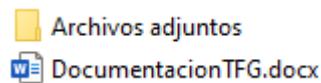


Figura 91 Archivos dentro de "Documentación App".

Dentro de la carpeta de archivos adjuntos se encuentran, como se ve en la Figura 92, los documentos referenciados en este documento:

- Una carpeta “Ayuda”, cuyo contenido se muestra en la Figura 93.
- Una carpeta “Planificación y presupuesto iniciales”, la cual contiene la planificación del proyecto estimada, realizada antes del desarrollo del mismo, y explicada en el Capítulo 3.
- El archivo de planificación, “Planificación.mpp”, el cual es modificable mediante “Microsoft Project”, y contiene un diagrama de Gantt con todas las tareas llevadas a cabo durante el proceso de desarrollo del proyecto, y se referencia en el Capítulo 10 de este documento.
- El archivo de presupuesto (Presupuesto.xlsx), modificable con el programa “Microsoft Excel”, el cual contiene el cálculo del presupuesto del proyecto.
- Dos vídeos en los que se muestra una demostración de el uso de las aplicaciones en 4 dispositivos Android al mismo tiempo.

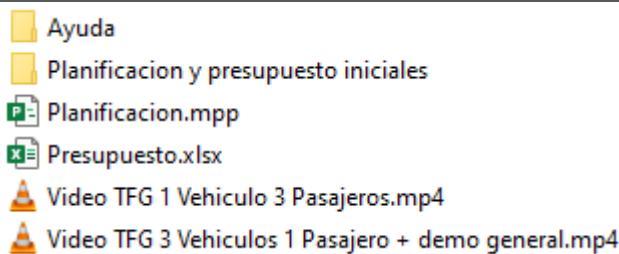


Figura 92 Contenido de la carpeta “Archivos adjuntos”.

En la Figura 93, observamos que en la carpeta “Ayuda”, se encuentran tres ficheros en formato PDF. Estos, contienen las guías de usuario de las aplicaciones, así como un mapa de zonas (utilizado en la aplicación del pasajero), el cual está basado en el mapa de zonas del Consorcio de Transportes de Asturias, proveniente de la siguiente referencia (CTA, 2022).

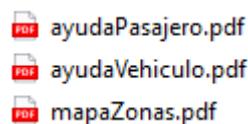


Figura 93 Contenido de la carpeta “Ayuda”.

13.2.1.2 Documentación Auditoría

Dentro de la carpeta Documentación Auditoría se encuentra, tanto el archivo Auditoría.pdf, como otras carpetas con diferentes archivos. La explicación de todas las carpetas y archivos presentes en la carpeta “Documentación Auditoría” se encuentra desglosada en el apartado “2.6. Fichero y tratamiento de objetos de la auditoría”, presente en el mismo archivo Auditoría.pdf.