CatTránsit. Aplicación del tráfico de Cataluña

Nombre Estudiante: Daniel Joares Torrado
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a: Albert Mata Guerra
Profesor/a responsable de la asignatura: Carles Garrigues Olivella

Fecha de Entrega 03/01/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*CatTránsit. Aplicación del tráfico de Cataluña*
Nombre del autor:	*Daniel Joares Torrado*
Nombre del consultor/a:	Albert Mata Guerra
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	01/2018
Titulación:	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	*Castellano*
Palabras clave	*Android, Open Data y REST*

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

La finalidad de este proyecto basado en el desarrollo de aplicaciones para dispositivos móviles es aumentar los conocimientos sobre el desarrollo en sistemas Android y utilizar una interfaz REST para recoger datos en un portal Open Data. Para ello, se creado una aplicación utilizando Android Studio y Retrofit como tecnologías principales.

A día de hoy, gracias a las nuevas tecnologías GPS y mapas, se ha mejorado la movilidad de las personas. Utilizando dichas tecnologías y datos abiertos al público se ha creado una aplicación para que los usuarios sepan en todo momento la situación del tráfico.

Para el desarrollo del proyecto se han seguido diferentes fases. Primero la búsqueda de un tema atractivo y posteriormente se empezó la fase de análisis de funcionalidades, requisitos y la planificación del proyecto. Una vez el análisis completado, se realizó el diseño de la aplicación. Definido el diseño se implementó la aplicación y finalmente se hicieron pruebas y la documentación pertinente del proyecto. Se ha tenido que volver al análisis de funcionalidades para replantear uno de los objetivos sin afectar el diseño. Se encontraron dificultades en la implementación ante nuevas situaciones que no se habían desarrollado anteriormente.

El resultado obtenido es el esperado salvo por un objetivo no cumplido que se tuvo que replantear si era necesario.

Como conclusión, el TFM me ha hecho aprender muchos conocimientos nuevos así como ampliar otros. Por otra parte, he podido desempeñar funcionalidades que no se había hecho antes como la planificación, y que es una parte imprescindible para el desarrollo de una aplicación.

Abstract (in English, 250 words or less):

The purpose of this Project based on the development of applications for mobile devices is to increase the knowledge on the development in Android systems and use a REST interface to collect data in an Open Data portal. For this, an application was created using Android Studio and Retrofit as main technologies.

Today, thanks to new GPS technologies and maps, people's mobility has been improved. Using these technologies and data open to the public, an application has been created so that users know the traffic situation at all times.

For the development of the project, different phases have been followed. First the search for an attractive topic and then analysis phase of functionalities, requirements and project planning began. Once the analysis completed, the design of the application was made. Once the design was defined, the application was implemented and finally the relevant tests and documentation of the Project were made. It has had to return to the analysis of functionalities to rethink one of the objectives without affecting the design. Difficulties were found in the implementation of new situations that had not been previously developed.

The result obtained is the expected one except for an unfulfilled objective that had to be reconsidered if necessary.

As a conclusion, the TFM has made me learn much new knowledge as well as expand others. On the other hand, I have been able to perform functions that had not been done before, such as planning, and which is an essential part of developing an application.

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Actualmente hay muchas aplicaciones que informan del estado de carreteras. Algunas muestran la densidad del tráfico, otras informan de sucesos en las carreteras, etc. por lo que se decide evolucionar un producto ya existente y mejorarlo.

Para cualquier conductor, yendo al trabajo o realizando un viaje de placer, encontrarse con atascos es una pesadilla. Moverse en transporte público, en bici o evitar las horas punta son algunas soluciones para evitar una congestión de tráfico. El uso de las nuevas tecnologías es una solución muy utilizada hoy en día.

Se decide desarrollar una aplicación basada en dispositivos Android con la que los usuarios puedan recibir información en tiempo real del tráfico en Cataluña. Pensada para mejorar la toma de decisiones de los usuarios cuando tengan que realizar un trayecto con sus vehículos y mejorar otras aplicaciones del mercado.

Algunas opciones no disponibles en ciertas aplicaciones se introducirán para mejorar la experiencia del usuario y obtener un producto lo más completo posible. Cabe destacar que la función principal de la aplicación es informar, no calcular rutas.

El objetivo es que las personas habituadas a conducir en su automóvil, sean particular o de empresa, pueda realizar con antelación una ruta que le permita llegar a su destino sin demoras. Y estar permanentemente informada de las noticias más recientes en las carreteras de Cataluña.

1.2. Objetivos del Trabajo

1.2.1. Objetivos

- Hacer un análisis correcto de los requisitos.
- Diseño agradable e intuitivo de las interfaces y realizar un diagrama de transiciones entre ellas coherente y de fácil uso.
- Estructurar bien la arquitectura de la aplicación.
- Implementación del producto utilizando API's y utilización de servidores externos mediante el entorno de desarrollo oficial de aplicaciones Android.
- Realización de pruebas.

1.2.2. Requerimientos funcionales

- Consultar el estado del tráfico en tiempo real.
- Mostrar la ubicación de los radares fijos.
- Mostrar la localización y visualización del estado del tráfico mediante cámaras ubicadas en determinadas vías.
- Listar las noticias recientes del servicio de tráfico catalán.
- Mostrar la ubicación del usuario en el mapa.
- Alertar al usuario de un imprevisto cerca de su ubicación.
- Guardar las preferencias del usuario.

1.2.3. Requerimientos no funcionales

- La aplicación debe ser fácil de utilizar.
- Las interfaces de la aplicación deben ser amigables e intuitivas.
- La aplicación debe de proporcionar tiempos de respuesta rápidos.
- La aplicación debe ser fácil de analizar y modificar para corregir posibles fallas.
- La aplicación debe ser fácil de descargar e instalar.
- La aplicación debe de proporcionar seguridad al usuario, es decir, no debe permitir que se instale algún otro software malicioso en la aplicación.

Se ha optado por un desarrollo exclusivo para la plataforma Android puesto que durante el máster se ha aprendido a utilizar el software para el desarrollo de aplicaciones para dicha plataforma y no para otras. Tampoco se dispone de las herramientas para desarrollar para otros sistemas como iOS. Esto facilita el desarrollo del TFM.

1.3. Enfoque y método seguido

Para el desarrollo del TFM se han seguido diferentes tareas. Primero pensar y buscar información sobre qué hacer la aplicación. Una vez decidido el tema, se hizo la planificación y se inició la primera fase del desarrollo de la aplicación, análisis y requerimientos funcionales. Terminada la primera fase se realiza el diseño de la aplicación, observar aplicaciones similares y estudiar a potenciales usuarios. Por último la fase de implementación y las pruebas.

El método de desarrollo a emplear es el Waterfall. El mero hecho de que no hay "cliente" con el que hablar de que es lo que quiere, los requisitos se especifican claramente y no van a cambiar durante el ciclo de vida del desarrollo, y si los hay se introducirán en futuras versiones. Por lo que no hay incertidumbre por lo que se desea hacer.

Se plantea realizar la aplicación utilizando el patrón arquitectónico MVC. Aislando las vistas de la lógica, y por otro lado las clases modelo que accederán a los datos a consultar mediante una base de datos. El controlador irá recibiendo las peticiones del usuario y devolviendo los datos a mostrar.

1.4. Planificación del Trabajo

1.4.1. Recursos

Para el desarrollo de la aplicación se ha utilizado el siguiente entorno de trabajo:

- Android Studio: IDE para el desarrollo de aplicaciones Android.
- Balsamiq Mockups 3: Aplicación para el diseño de pantallas de bajo nivel.
- Draw.io: Aplicación web para realizar diagramas.
- Open Data Generalitat: Web para la obtención de los datos necesarios.
- GitLab y Google Drive: Control de versiones del proyecto.
- GoogleMaps: Api para construir y personalizar el mapa de tráfico.
- Retrofit: librería para realizar peticiones Http en aplicaciones Android.
- JCoord: librería para realizar la conversión de coordenadas UTM a grados.

1.4.2. Tareas

La planificación está basada en las fechas de entrega de las diferentes PAC. El tiempo dedicado es una aproximación que puede alterarse debido a las dificultades que se puedan encontrar durante el desarrollo del proyecto. Se calculan unas 3 horas diarias que pueden variar según las dificultades las dificultades que puedan surgir durante el proyecto.

Tarea	*Inicio*	*Fin*	*Horas*
Plan de trabajo			
Búsqueda del temario	20/09/17	03/10/17	20
Análisis de requerimientos	03/10/17	09/10/17	10
Planificación del proyecto	09/10/17	11/10/17	4
Diseño			
Usuarios y contexto de uso	12/10/17	12/10/2017	3
Diseño conceptual	13/10/17	13/10/17	3
Prototipo	14/10/17	21/10/17	20
Evaluación	22/10/17	23/10/17	5
Diseño técnico	24/10/17	01/11/17	20
Implementación			
Preparación del entorno de trabajo y descarga de librerías	02/11/17	02/11/17	3
Desarrollo de la aplicación	03/11/17	06/12/17	150
Pruebas	07/12/17	13/12/17	20
Entrega final			
Memoria del proyecto	03/10/17	13/12/17	50
Presentación	14/12/17	03/01/18	20

Plan de trabajo

La primera fase del proyecto es la fase más importante del proyecto donde se realiza el análisis de requerimientos y la planificación de todo el TFM. Antes de empezar, se hizo una búsqueda de un temario atractivo.

La fase inicial consiste en tres partes:

- Búsqueda del temario: Como no había un tema definido para realizar el TFM, durante dos semanas se realizó una búsqueda de un temario que fuese motivador. Se decidió que este estuviese relacionado con la obtención de datos abiertos. Se llegó a replantear tres veces el temario a realizar.

- Análisis de requerimientos: Una vez decidido el tema a tratar, en los siguientes días se realiza un estudio del producto que queremos obtener. ¿Qué debe hacer? ¿Hay otras aplicaciones similares? ¿A quién va dirigido?

- Planificación del proyecto: Una vez se tiene claro lo que se quiere, se realiza la planificación del proyecto haciendo una estimación aproximada de las horas dedicadas a este y las tareas a cumplir en cada fase. Aunque termina la primera fase esta puede verse modificada a lo largo del proyecto y verificar que se cumplen las fechas establecidas para completar cada tarea.

En la primera fase se cumplen las tareas establecidas en el período asignado.

Diseño

La segunda fase planteamos el diseño de la aplicación, de duración tres semanas, se realizan las siguientes tareas:

- Usuarios y contexto de uso: Se hace un estudio de que usuarios pueden beneficiarse de la aplicación y en qué casos puede utilizarla. Se entrevista a dos personas conocidas que son potenciales usuarios. La tarea se cumple en un día.

- Diseño conceptual: Al día siguiente se replantean las pantallas a desarrollar y como el usuario navegará por ellas.

- Prototipo: Durante una semana se realizan los esbozos de cada una de las pantallas así como replantear las funcionalidades que deben realizar cada una de ellas.

- Evaluación: Se estudia cada uno de los prototipos de pantalla si cumplen los principios de usabilidad (heurísticas).

- Diseño técnico: La última semana se especifican los casos de uso, que tipo de datos necesitamos obtener y cómo vamos a obtenerlos diseñando la arquitectura de la aplicación.

No hay demoras durante la fase de diseño.

Implementación

Durante seis semanas se desarrolla la aplicación aplicando el diseño de la anterior fase. Esta fase consiste en las siguientes tareas.

- Preparación del entorno: Se prepara el IDE y se descargan las librerías que se utilizarán durante el desarrollo.

- Desarrollo de la aplicación: El desarrollo de la aplicación se prolonga a lo largo de cinco semanas. Se empieza desarrollando las interfaces siguiendo los prototipos realizados en el diseño. Se estructura la API Rest para la obtención de datos. Y por último se desarrollan las diferentes pantallas de la aplicación.

- Pruebas: La última semana se realizan las pruebas conforme todo funciona correctamente.

En la fase de implementación se replanteo uno de los objetivos, si era necesario enviar notificaciones al usuario. Esto no afectó ni al diseño ni a la planificación que siguió tal y como se planteó de inicio.

Entrega final

La última fase del proyecto tiene una duración de tres semanas para terminar el TFM. Estas son las tareas a realizar:

- Memoria del proyecto: En cada una de las anteriores fases se ha ido cumplimentando la memoria todo lo que se trabajaba, pero no es en la última fase donde se realizan los retoques finales y correcciones.

- Presentación: Primero se plantea cómo se presentará el proyecto y qué información se divulgará. Para no improvisar se escribe un monólogo. Por último se graba un vídeo de la demo de la aplicación y de la presentación.

Durante esta fase se corrigieron algunos errores menores de la aplicación antes de su entrega.

1.5. Breve sumario de productos obtenidos

La entrega final del proyecto incluye los siguientes elementos:

- La aplicación: incluye todos los archivos necesarios para su funcionamiento.
- La memoria del proyecto: explicación de todas las tareas realizadas durante el proyecto y un manual de uso de la aplicación.
- Archivo ejecutable de la aplicación.
- La presentación: vídeo explicativo del proyecto y el funcionamiento del producto.

1.6. Breve descripción de los otros capítulos de la memoria

A continuación se detalla el contenido de cada uno de los capítulos referente al desarrollo de la aplicación.

En el capítulo de diseño se explicarán los siguientes puntos:

- A quién va dirigido la aplicación, y en qué caso les puede ser útil.
- Qué diseño conceptual se optó por realizar y porqué. Como van interactuar las actividades entre sí.
- Muestra de los esbozos del prototipo de bajo nivel de cada pantalla y explicación detallada de sus funcionalidades.
- Cumplen los principios de usabilidad las interfaces? Se realiza una evaluación heurística para llevar a cabo la tarea.
- Cuáles casos de uso hay, qué puede realizar el usuario.
- Diseño de la arquitectura de la aplicación y como obtendremos los datos.

El contenido del capítulo de implementación es el siguiente:

- Arquitectura del proyecto
- explicará el entorno de trabajo empleado, posibles mejoras de la aplicación y las conclusiones.
- Que librerías se ha utilizado y cómo se integran en el proyecto.

El capítulo de futuras versiones se detalla las mejoras y correcciones que se introducirán en siguientes versiones de la aplicación.

Los capítulos Glosario y Bibliografía hay listados los términos utilizados en la memoria y recursos utilizados durante el proyecto como webs y artículos.

El último capítulo, Anexos, tenemos un manual de uso de la aplicación y las instrucciones de su instalación.

2. Diseño

2.1. Usuarios y contexto de uso

2.1.1. A qué tipo de usuario va dirigido

A día de hoy circular por la ciudad, la autopista, etc. puede ser estresante. El tránsito puede ser denso a ciertas horas del día, por lo que sería ideal planear un recorrido para llegar a tiempo a una cita. Por lo tanto, para muchos conductores es primordial llegar puntual al trabajo o viajeros que quieren evitar demoras hacia su destino requieren de la información necesaria para poder planificar otras rutas. La gente que utiliza el transporte público y sabe de antemano que puede retrasarle el viaje, le permite buscar otras alternativas.

2.1.2. Fichas de personas y escenario

En este apartado se explican dos de los casos de uso más importante, puesto que en cada uno de ellos se pueden acceder al resto de funcionalidades que son complementarias.

Nombre: Sofía
Edad: 28 años
Profesión: administrativa
Descripción de la persona:

Sofía vive junto con su marido en Barcelona. Se desplaza para trabajar como recursos humanos de lunes a viernes durante una jornada de ocho horas.

Le encanta pasear, ir de compras y salir con los amigos. De vez en cuando le apetece salir algún fin de semana de viaje. Los días laborales en sus ratos libres suele ver series o utilizar su tableta para leer o jugar. Aunque tiene ordenador, no suele utilizarlo a menudo, solo su marido.

Barcelona es una ciudad muy problemática en cuanto al tráfico, sobretodo en horas puntas. Aunque tiene coche, es su marido quien lo utiliza para desplazarse a trabajar fuera de Barcelona y ella se desplaza en autobús ya que la línea que pasa cerca de su casa le deja cerca del trabajo. Sin embargo, el trayecto no es corto, y hay peligro de que llegue tarde al trabajo.

Descripción del escenario:

Sofía y su marido tienen previsto salir de viaje el fin de semana, así que el viernes por la noche, Sofía abre la aplicación para ver las noticias y ver cuál es la previsión en las carreteras catalanas. Se levanta el sábado a las 9.00h, para desayunar y arreglarse antes de salir de casa y coger el vehículo. Mira el mapa del tránsito y observar cómo están las vías menos congestionadas y si hay alguna incidencia.

Nombre: Andreu
Edad: 32 años
Profesión: programador
Descripción de la persona:

Andreu vive con sus padres en Mataró. La empresa en la que trabaja reside en Barcelona, así que todos los días laborales se tiene que desplazar con su vehículo por la autopista.

Andreu en sus ratos libres le gusta jugar a los videojuegos o ver series con su novia durante la semana. Los fines de semana los pasa con sus amigos. Además se ha hecho youtuber, y publica de vez en cuanto algún vídeo porque le gusta, no por profesión.

El tráfico a Barcelona suele ser denso, y muchas veces, llegando a la ciudad, las congestiones son habituales. Aunque es previsor y sale de casa con tiempo a veces no es suficiente. Siempre utiliza la autopista y las calles de Barcelona más directas a su trabajo porque es el camino más corto pero puede que no sea el más rápido.

Descripción del escenario:

El miércoles por la noche, Andreu mira su Smartphone y mira las noticias en su aplicación por si puede ver alguna incidencia mañana de camino a su trabajo. Andreu se levanta el jueves a las 6.45h. Desayuna, se viste y se monta en su coche para dirigirse al trabajo. Antes de poner en marcha el coche mira su Smartphone para informarse como están las carreteras, mira si hay incidencias que puedan afectarle y decide que ruta realiza.

2.2. Diseño conceptual

Haciendo repaso de las funcionalidades que debe realizar la aplicación las podemos agrupar en 6 pantallas. A partir de la pantalla principal el usuario puede desplazarse a otras pantallas de la siguiente manera. En la parte superior derecha de la pantalla se mostrará siempre el menú al usuario. Desde el menú se accederá al resto de pantallas e igualmente desde cada una de ellas al resto. Teniendo libertad para desplazarse por cualquier de las funcionalidades que la aplicación ofrece en todo momento. En la figura inferior muestra el árbol de navegación de la aplicación.

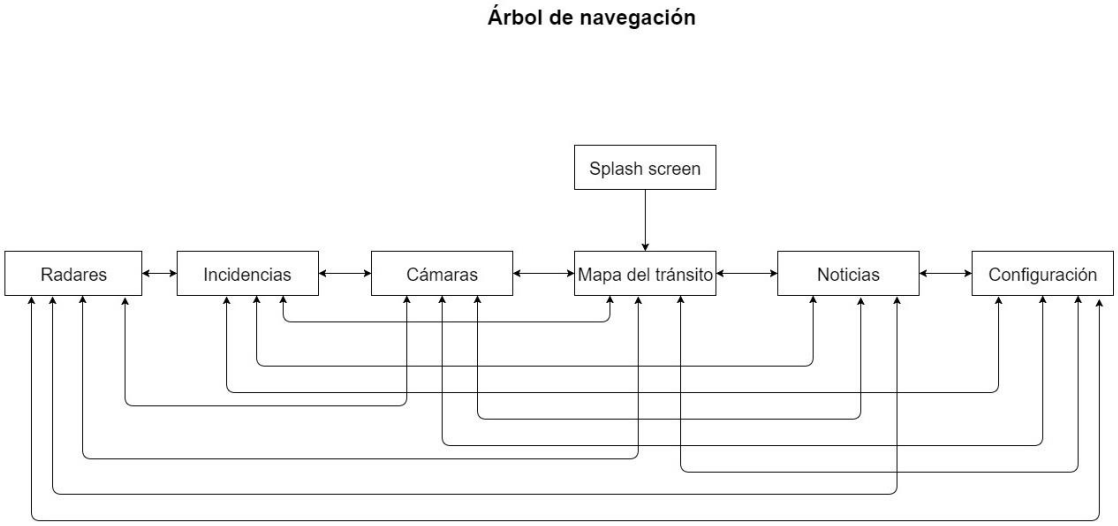

Ilustración 1: **Árbol de navegación de la aplicación.**

Un motivo por el que se ha decidido esta navegación es por su sencillez y rapidez para moverse por las diferentes pantallas. En algunas aplicaciones el usuario está obligado a regresar a la pantalla principal para volver de nuevo al menú para desplazarse de nuevo. Esto puede ser engorroso. Por este motivo, pulsando el botón del menú que tiene cada interfaz el usuario tiene total libertad a donde quiere dirigirse.

Porque navegar utilizando un botón en vez deslizar la pantalla que puede ser más ágil. En las pantallas de la aplicación se muestran elementos que al pulsar pueden mostrar información o enviar al usuario a un enlace exterior. No es agradable que al intentar desplazarte se muestre sin querer información no solicitada. No todo el mundo tiene la habilidad de utilizar dispositivos móviles sin complicaciones y aun así puede ocurrir este suceso.

2.3. Prototipo

En este apartado se realizará un repaso de los prototipos de bajo nivel de cada una de las pantallas, y se explicará los motivos del diseño elegido.

La Splash screen es la pantalla que aparece al iniciar la aplicación. Se muestra el título de la aplicación y el logo de la Generalitat de Catalunya, que por temas de licencia de datos, tiene que hacer referencia la fuente de la información reutilizada para el desarrollo de la aplicación. Tras unos segundos se iniciará la actividad principal.

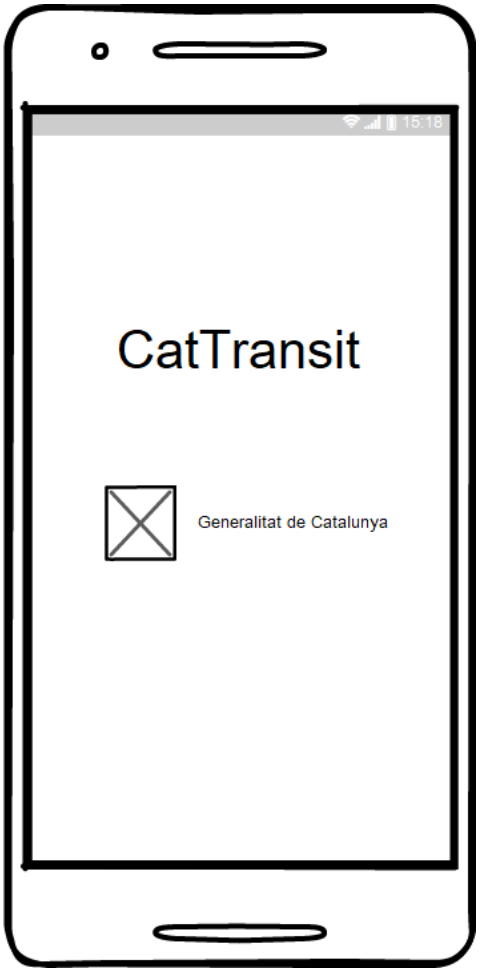

Ilustración 2: Esbozo de la interfaz de inicio de la aplicación.

La actividad principal se centra en el mapa del tránsito que con la colaboración de Google Maps mostrará la densidad del tráfico. En la parte superior se muestra una barra con el icono del menú, el título de la pantalla, el buscador y el botón de actualizar. Las personas leemos de izquierda a derecha y de arriba hacia abajo, y lo mismo para leer pantallas. El usuario lo primero que verá será la barra con estos elementos. Por eso se decide optar en poner el menú en la parte superior izquierda. Este patrón se repite en cada una de las diferentes pantallas salvo por el buscador que es exclusivo del mapa y el botón de actualización que en la pantalla de configuración no tiene sentido.

Ilustración 3: Esbozo de la interfaz del mapa de tráfico.

En la pantalla del mapa, si el usuario tiene activado la visualización de elementos, puede ver diferentes símbolos representando incidencias, cámaras, radares, etc. Pulsando encima de ellos se desplegará en la parte inferior un cuadro con información de dicho elemento. Sin acumular todo en la parte superior de la pantalla. Para el caso de uso de Andreu, visualizar el mapa con todos los detalles, le permite al momento decidir que ruta elegir.

Ilustración 4: Esbozo de la interfaz del mapa mostrando el menú desplegable y la información de una incidencia.

En la pantalla de incidencias se muestra un listado de todas las incidencias de Cataluña al momento, indicando la información de cada una de ellas. Estas también aparecen en el mapa en forma de iconos como previamente se ha descrito. El usuario puede deslizar hacia abajo para ver la lista completa.

El usuario pude seleccionar la incidencia para acceder a más información en la web del SCT.

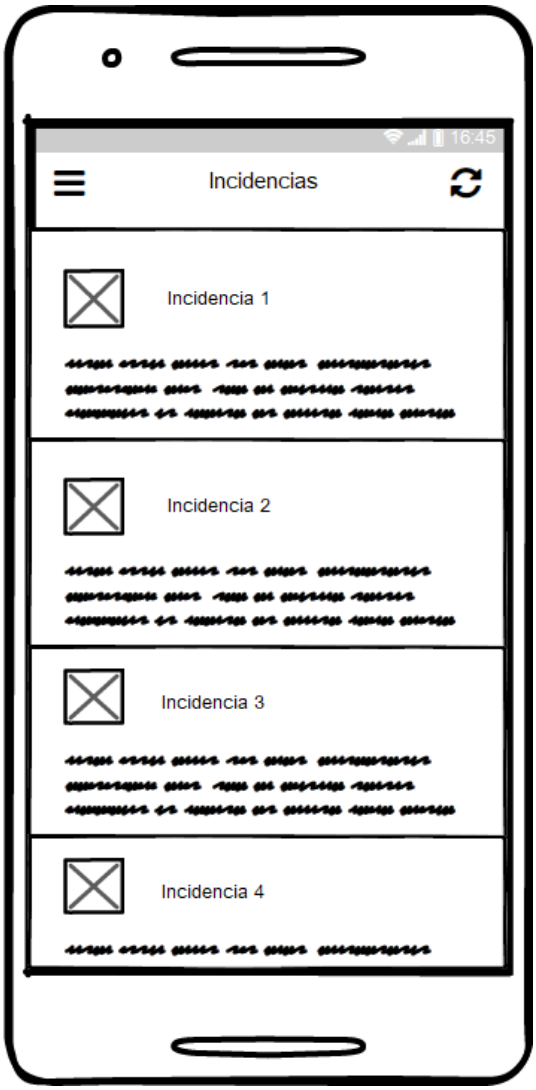

Ilustración 5: Esbozo de la interfaz de incidencias

La pantalla de cámaras es muy similar a la de incidencias. Un listado con todas las cámaras, mostrando la imagen y la información de cada una de ellas. Permite echar un vistazo rápido de la situación de diferentes puntos de las carreteras en Cataluña.

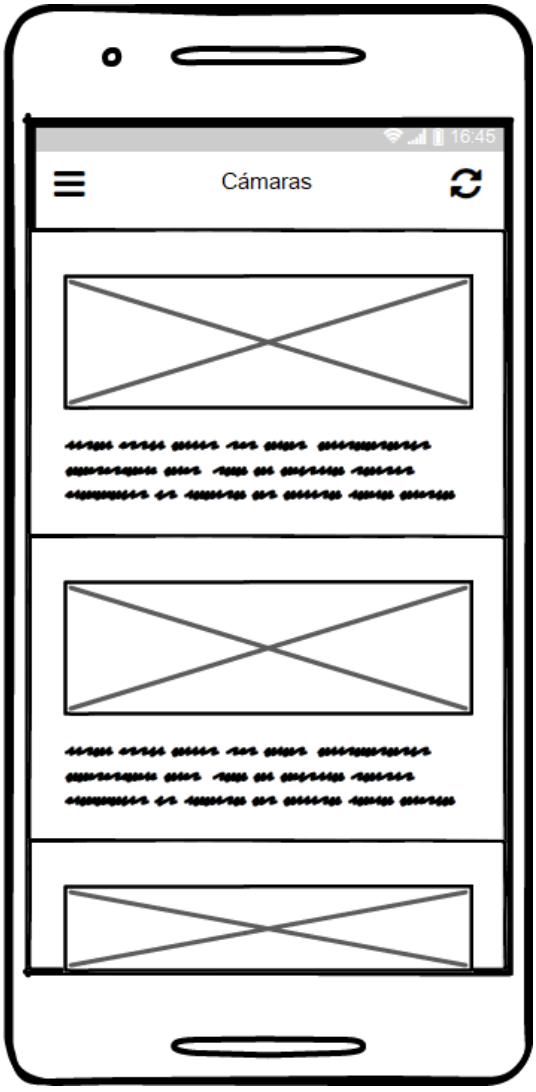

Ilustración 6: Esbozo de la interfaz de cámaras

El listado de noticias sigue el mismo patrón que las anteriores pantallas de listado, y su funcionamiento para moverse entre ellas. En el caso de uso de Sofía, puede visualizar las últimas noticias y prever imprevistos. El título de la noticia está vinculado a la página de noticias del Servei Català de Trànsit, donde se especifica toda la información.

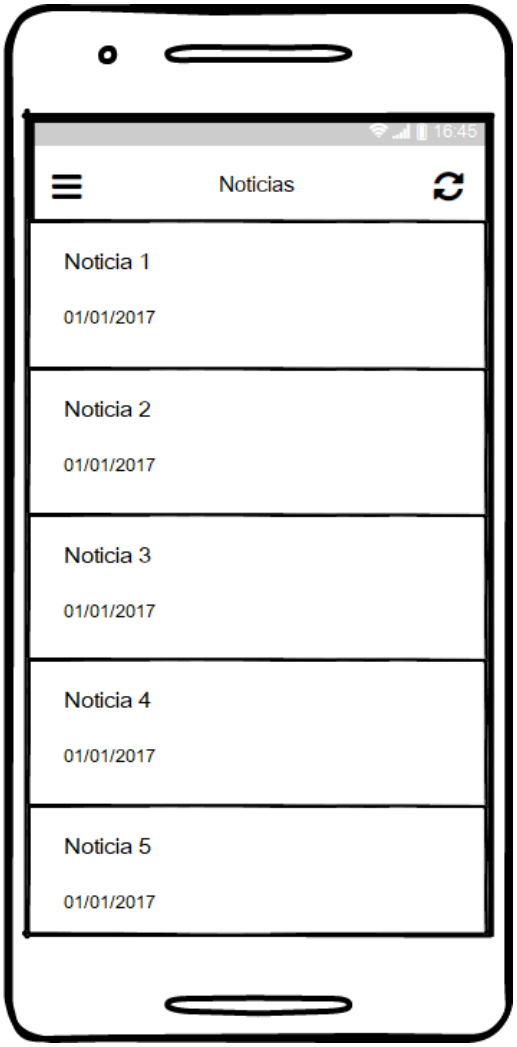

Ilustración 7 Esbozo de la interfaz de noticias

La pantalla de radares detalla cada uno de estos con su información en una lista. Como previamente hemos visto en las pantallas de incidencias y cámaras el usuario puede deslizar la pantalla hacia abajo para observar el listado.

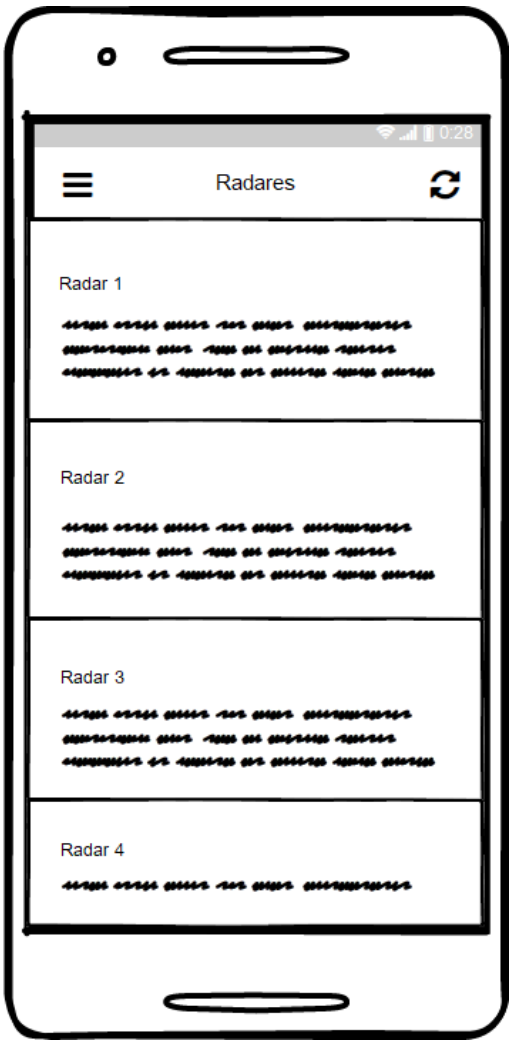

Ilustración 8: Esbozo de la interfaz de radares

Por último, la pantalla de configuración es muy sencilla. El usuario dispone un desplegable para elegir el idioma que desee utilizar en la aplicación y la opción de mostrar los elementos en el mapa. De momento solo se contempla estas dos opciones, en futuras mejoras se añadirán más si es necesario.

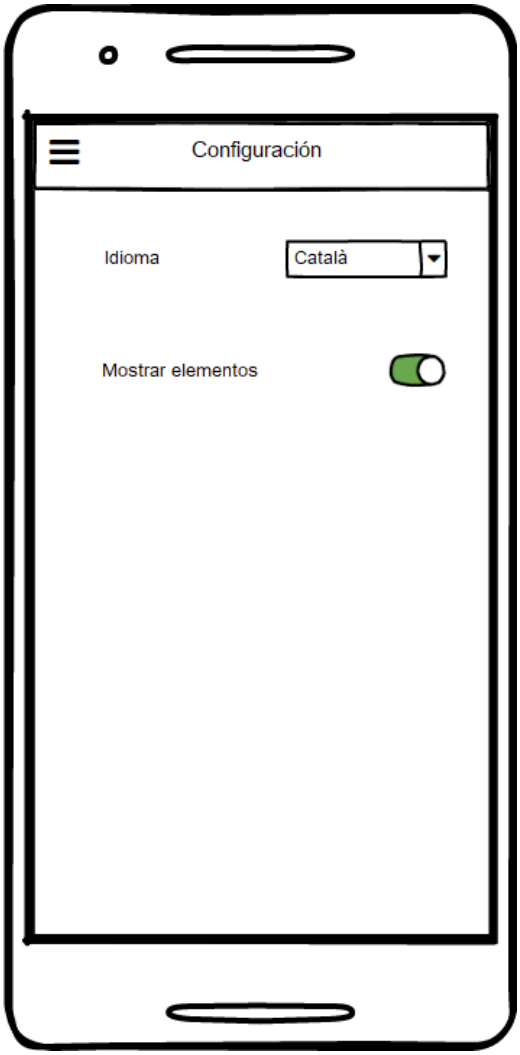

Ilustración 9: Esbozo de la interfaz de configuración

2.4. Evaluación

Se decidió utilizar la evaluación heurística porque es un método que examina las interfaces y se determina si cumplen los principios de usabilidad. Se recorre la interfaz varias veces, examinando cada elemento y los comparamos con los principios (heurísticas).

Visibilidad del estado del sistema: El sistema obtiene la información en tiempo real, gracias a los datos aportados por la Generalitat de Cataluña, por lo que usuario está informado en todo momento de cualquier suceso. Es posible en el caso de las pantallas de cámaras que estas no se obtengan imágenes actualizadas constantemente y en algún caso ni siquiera se obtiene dicha imagen por problemas técnicos.

Consistencia entre el sistema y el mundo real: El sistema utiliza un lenguaje que el usuario le resulta familiar. Además dispone de cambiar de idioma si lo desea.

Control de usuario: El usuario tiene la capacidad de cambiar en cualquier momento de pantalla y elegir la que desee o abandonar la aplicación, en otras palabras, tiene el control. En una situación no deseada, como la elección de una noticia erróneamente, le lanzará a la página de noticias del SCT, el usuario puede retornar atrás.

Consistencia y estándares: La aplicación se desarrolla para dispositivos Android, por lo que el sistema sigue las normas de la misma plataforma. El usuario que utilicé estos dispositivos podrá afrontar acciones del sistema.

Prevención de errores: El sistema previene de la existencia de errores, por ejemplo, en el caso de no obtener la imagen de una cámara, el sistema informa al usuario.

Minimizar la carga de la memoria del usuario: Se utiliza un sistema de navegación en la parte superior de cada una de las pantallas con la misma simbología, para que el usuario no deba recordar los detalles de navegación. Los iconos son fáciles de reconocer su utilidad. El mapa dispone de iconos que representa señales de tráfico correspondientes a la información que detallan.

Flexibilidad y eficiencia de uso: El diseño de las diferentes pantallas es sencillo y práctico, contienen la información necesaria y en el caso del mapa, si este no visualiza todos los elementos el usuario puede navegar a otra página y ver el listado de estos. De fácil de uso, tanto para usuarios nuevos como para los experimentados.

Diseño práctico y sencillo: El sistema de navegación es sencillo y su interfaz es sencilla y contiene la información relevante.

Ayudar a los usuarios a reconocer, diagnosticar y deshacer errores: Los mensajes de error se expresan con un lenguaje claro, indicando el problema.

Ayuda y documentación: Disponer ayuda es importante, pero mejor si el usuario no la necesita. En la aplicación no existe documentación relativa al funcionamiento de la aplicación. No se descarta añadirla más adelante.

2.5. Diseño técnico

2.5.1. Diagrama de casos de uso

En el siguiente diagrama se pueden ver los casos de uso.

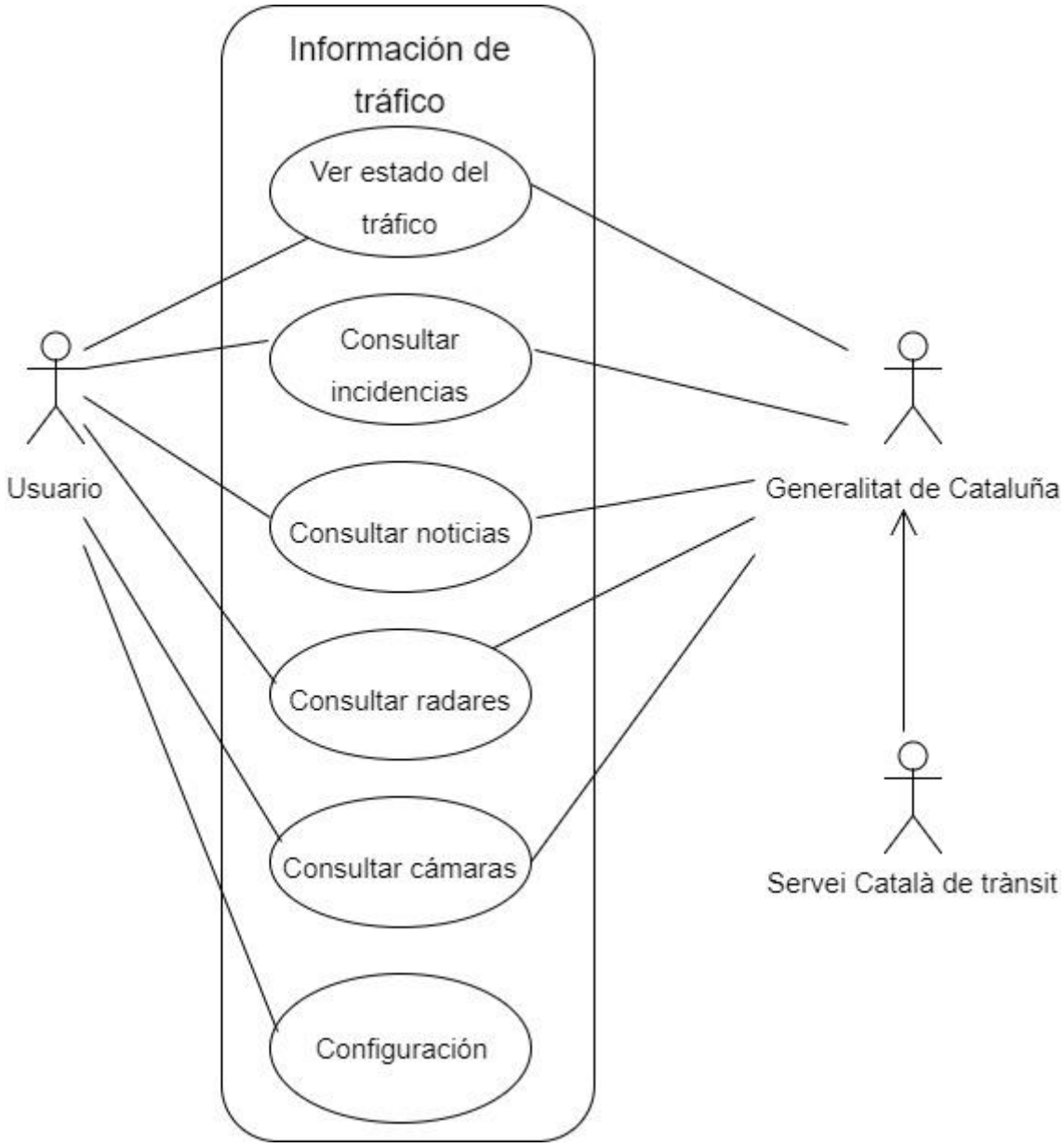

Ilustración 10: Diagrama de casos de uso

2.5.2. Especificación de casos de uso

Ver estado del tráfico		
Roles	Cualquier usuario	
Condición previa	Ninguna.	
Secuencia principal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El sistema muestra la pantalla del mapa.
Alternativas	Paso	Acción
	2	No hay conexión a internet y no muestra los datos en tiempo real.

Consultar incidencias		
Roles	Cualquier usuario	
Condición previa	Ninguna.	
Secuencia principal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El sistema muestra la pantalla del mapa.
	3	El usuario pulsa el botón menú y elige “Incidencias”.
	4	El sistema muestra la pantalla de incidencias.
	5	El usuario puede acceder a más información pulsando en cualquiera de ellas.
	6	El sistema le reenvía a la web del SCT.
Alternativas	Paso	Acción
	4	No hay conexión a internet y no muestra los datos actualizados.

Consultar noticias		
Roles	Cualquier usuario	
Condición previa	Ninguna.	
Secuencia principal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El sistema muestra la pantalla del mapa.
	3	El usuario pulsa el botón menú y elige “Noticias”.
	4	El sistema muestra la pantalla de noticias.
	5	El usuario puede acceder a más información pulsando en el enlace de la noticia.
	6	El sistema le reenvía a la web del SCT.
Alternativas	Paso	Acción
	4	No hay conexión a internet y no muestra los datos actualizados.

Consultar radares		
Roles	Cualquier usuario	
Condición previa	Ninguna.	
Secuencia principal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El sistema muestra la pantalla del mapa.
	3	El usuario pulsa el botón menú y elige “Radares”.
	4	El sistema muestra la pantalla de radares.
	5	El usuario puede ver el listado de radares.
Alternativas	Paso	Acción
	4	No hay conexión a internet y no muestra los datos actualizados.

Consultar cámaras		
Roles	Cualquier usuario	
Condición previa	Ninguna.	
Secuencia principal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El sistema muestra la pantalla del mapa.
	3	El usuario pulsa el botón menú y elige “Cámaras”.
	4	El sistema muestra la pantalla de cámaras.
	5	El usuario puede ver el listado de cámaras.
Alternativas	Paso	Acción
	4	No hay conexión a internet y no muestra los datos actualizados.

Configuración		
Roles	Cualquier usuario	
Condición previa	Ninguna.	
Secuencia principal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El sistema muestra la pantalla del mapa.
	3	El usuario pulsa el botón menú y elige “Configuración”.
	4	El sistema muestra la pantalla de configuración.
	5	El usuario puede modificar las opciones que se presentan.
Alternativas	Paso	Acción
		Ninguna.

2.5.3. Modelo Conceptual

Observamos las clases del modelo conceptual de la aplicación.

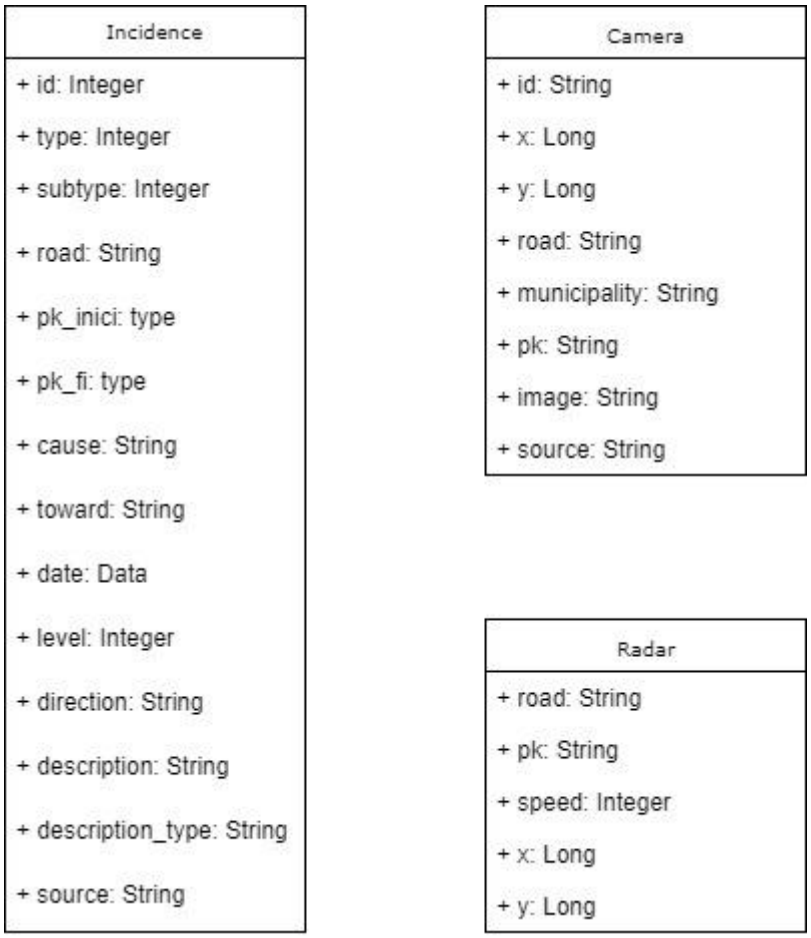

Ilustración 11: Modelo conceptual

2.6. Obtención de datos

En la página de datos abiertos de la Generalitat de Cataluña se obtendrá la información necesaria para mostrar los datos en la aplicación. Se requieren datos de las incidencias, cámaras, radares de la comunidad catalana. Estos datos están los siguientes formatos: xml, txt.

Aquí podemos observar la arquitectura que tomará.

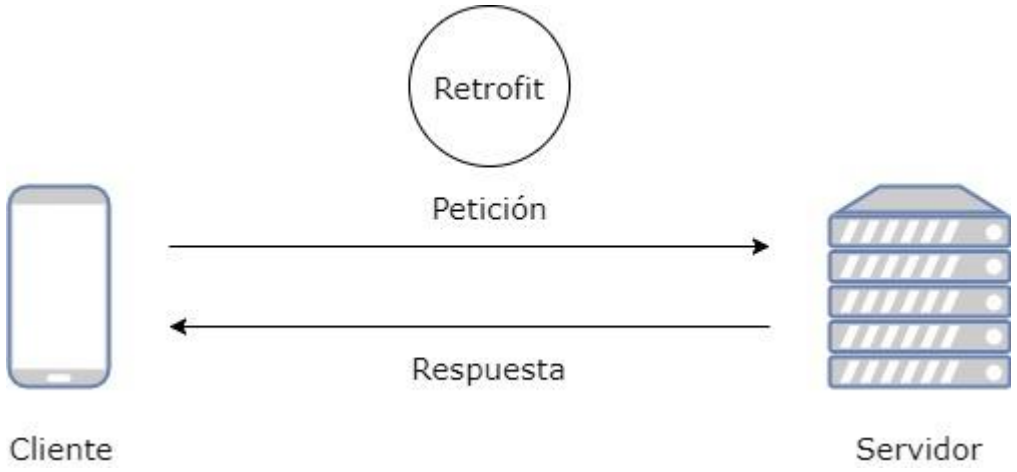

Ilustración 12: Modelo MVC de red

La aplicación debe realizar peticiones hacia el servidor para obtener los recursos xml y luego presentar la información al usuario, se puede considerar un diseño Modelo Vista Controlador de Red.

El diagrama muestra como la actividad realiza una petición con Retrofit hacia el servidor, la cual enviará una respuesta. Luego se actualiza la vista.

Desde la actividad, el controlador estará pendiente para mostrar el detalle de cada elemento.

Los datos de los radares se obtienen en formato txt, distinto a xml y se deberá realizar un tratamiento distinto para recuperarlos. Se plantean dos opciones:

1. Descargar el fichero .txt y extraer la información del fichero.

2. Convertir los datos del fichero .txt a JSON. Este fichero se puede almacenar en Firebase, y descargar estos datos para serializarlos en un objeto.

En el capítulo de implementación se detallará con más detalle los procedimientos y recursos utilizados para recuperar los datos.

3. Implementación

3.1. Arquitectura

En este punto se explicará la arquitectura utilizada en el proyecto y el diseño de la misma.

La aplicación se ha creado siguiendo el patrón arquitectónico MVC, utilizando las librerías Retrofit y GoogleServices. El proyecto se ha dividido en tres capas.

1. Presentación: En esta capa de la aplicación, la aplicación interactúa con el cliente. Se visualizan las funciones de la aplicación y se recogen y se muestran los datos. Con el IDE Android Studio se diseña las pantallas y la lógica para comunicarse con la capa.

2. Servicio: En esta parte se reciben los datos, se interactúa con la API REST utilizando la librería Retrofit, que realiza las peticiones al servidor y devuelve los resultados.

3. Datos: Es donde se almacenan los datos. Al tratarse de un proyecto Open Data, los datos están disponibles de forma libre para todo el mundo en un servidor externo por lo que no hay que implementar ninguna base de datos.

El módulo principal del proyecto ficheros se divide en tres partes. Primero la carpeta “manifests”, contiene la definición en XML muchos de los aspectos principales de la aplicación, como permisos y componentes (pantallas, servicios,…). La carpeta “java”, que contiene el código fuente de la aplicación, clases auxiliares, etc. La carpeta “res”, esta almacena los recursos necesarios para el proyecto: imágenes, layouts, cadenas de texto, etc.

No menos importante es el módulo Gradle Scripts. En él se destaca el fichero build.gradle, que contiene la información necesaria para la compilación del proyecto, por ejemplo la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a las librerías externas utilizadas, etc.

En un mismo proyecto pueden existir varios ficheros build.gradle, para definir determinados parámetros a distintos niveles. En la ilustración 13 se observa que hay un fichero build.gradle a nivel de proyecto, y otro a nivel de módulo dentro de la carpeta /app. El primero de ellos definirá parámetros globales a todos los módulos del proyecto, y el segundo sólo tendrá efecto para cada módulo en particular.

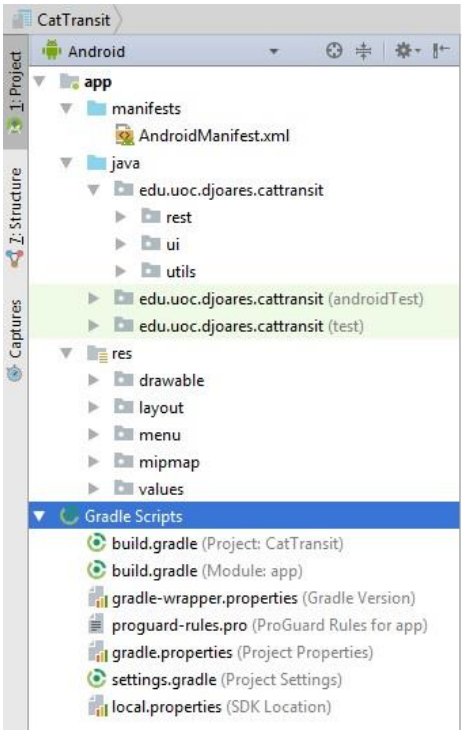

Ilustración 13: Estructura del proyecto

A continuación se explicará con más detalle cada una de las subcarpetas del módulo principal app.

3.2. Carpeta manifests

El fichero AndroidManifest.xml es un componente importante en cualquier proyecto de Android.

En este fichero se definen las actividades que en caso contrario al acceder a estas la aplicación falla.

También se declaran todos los permisos disponibles para la aplicación. Para este proyecto se ha declarado para acceder a Internet, localización del dispositivo, poder utilizar los servicios de Google y la llave de Google Maps v2.

Ejemplos:

Permiso para acceder a Internet:

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

Defínción de una actividad, en este caso la que arrancará la aplicación.

```
<activity
 android:name=".ui.activities.SplashActivity"
 android:theme="@style/SplashTheme">
 <intent-filter>
     <action android:name="android.intent.action.MAIN" />

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

Llave Google Maps v2:

```
<meta-data
 android:name="com.google.android.geo.API_KEY"
 android:value="@string/google_maps_key" />
```

3.3. Carpeta java

La carpeta java se compone de tres subcarpetas:

Rest: implementación de la interfaz REST
Ui: desarrollo de la lógica de las interfaces
Utils: se guardan las preferencias del usuario

3.3.1. Carpeta rest

La estructura de la carpeta "rest" está compuesta por cuatro subcarpetas como se observa en la siguiente ilustración:

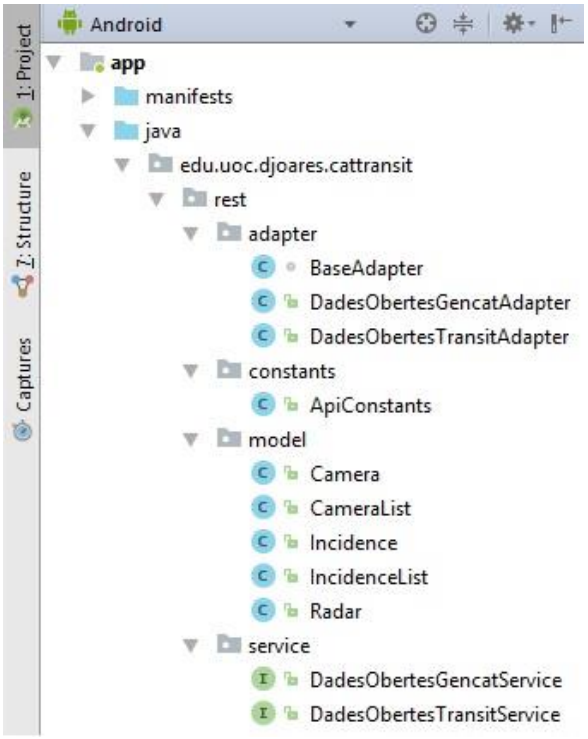

Ilustración 14: Detalle de la carpeta rest

La carpeta "constants" contiene la clase ApiConstants, donde se definen las constantes de la API de les dades obertes de la Generalitat. Estas constantes son las URL que se conectará la REST.

Le sigue la carpeta "service", se implementan las interfaces con las consultas REST que queremos realizar. En el proyecto realizaremos dos consultas en la interfaz "DadesObertesGencatService" para obtener el listado de cámaras e incidencias. En la interfaz "DadesObertesTransitService" realizaremos la consulta para obtener los radares.

En la carpeta "adapter" hay la entidad "BaseAdapter" que inicializa el cliente de la librería de Retrofit con OkHttp. Las otras dos clases "DadesObertesGencatAdapter" y

DadesObertesTransitAdapter" son clases que heredan los métodos y atributos de la entidad "BaseAdapter" e implementan las interfaces la carpeta service.

Por último, la carpeta "model" contiene el modelo de la respuesta que devuelve la API del usuario. Anteriormente se especificó que solo tres elementos formaban parte del modelo conceptual (cámara, radar e incidencias). Si nos fijamos, en la carpeta hay cinco entidades, se han añadido CameraList e IncidenceList. Los datos de estos elementos están en formato XML, y para deserializarlos es conveniente crear una instancia donde se guarde el listado de los elementos y por otro lado la instancia de dicho elemento donde se especifican los atributos que queremos obtener.

En el caso de los radares, la información esta almacenada en un archivo de texto que para convertir dicha información en un objeto se utilizará una clase de la carpeta "utils" para realizar la conversión.

3.3.2. Carpeta ui

En Android, las interfaces con los usuarios se crean utilizando layouts, que más adelante se explican. Todos los layouts implementados al proyecto se ajustan a las necesidades de las diferentes pantallas diseñadas de la aplicación.

Una vez implementado el layout, creamos su Activity asociada. Las Activities definen el comportamiento de cada uno de los componentes definidos en los layouts.

Las activities del proyecto se crean partir de clases java. En la imagen inferior se observa la estructura de la carpeta ui con las diferentes clases que implementan los layout.

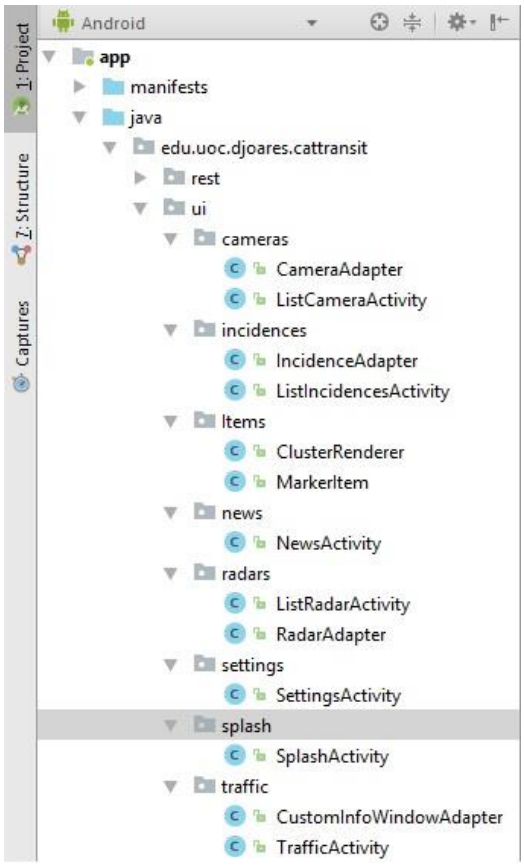

Ilustración 15: Detalle de la carpeta ui

Todas la Activities implementadas en el proyecto permiten dar el comportamiento lógico requerido a la aplicación.

Revisemos cada una de las activities.

Nota: En el anexo del documento se ha añadido un manual de usuario donde también de detallan las pantallas y sus funciones con sus correspondientes imágenes.

Activity: SplashActivity.java

Actividad de inicio de la aplicación. Muestra el título de la aplicación, el logo de la Generalitat de Catalunya y el logo del Trànsit de Catalunya. Estos últimos se incluyen porque los datos se obtienen de un portal de datos abiertos y se debe citar las fuentes de origen. La actividad se mostrará durante un solo segundo antes de avanzar a la actividad de tráfico para no sea molesto para el usuario.

Activity: TrafficActivity.java

Actividad donde el usuario puede visualizar el mapa de tráfico y los diferentes elementos como incidencias o radares si estos están habilitados.

Para la visualización del mapa se utiliza la API GoogleMaps. Gracias a ella se puede habilitar la capa de tráfico y ver la congestión de las carreteras mediante un código de colores. También incluye el botón de localización que sirve para mover la cámara hasta la posición del dispositivo. Se ha implementado un segundo botón personalizado de información para mostrar una leyenda mostrando los diferentes iconos que puede encontrar el usuario en el mapa.

Sin embargo, estos iconos pueden estar muy juntos y mostrarse apilados unos encima de otros si el zoom del mapa está demasiado lejos. Para solucionar este problema se utiliza la utilidad de agrupación de marcadores en clústeres, agregando marcadores en objetos ClusterItem al ClusterManager.

En la ilustración 15 se observa en la subcarpeta items dos clases, MarkerItem.java y ClusterRenderer.java. La primera sirve para personalizar el marcador, en el proyecto mostrar el icono correspondiente al elemento a mostrar y la segunda clase se encarga de la representación, agrega y elimina marcadores individuales y clústeres.

Por último la clase CustomInfoWindowAdapter.java que sirve para personalizar las ventanas de información emergentes al seleccionar un marcador. Para ello se utiliza el mismo layout que en los listados que veremos a continuación.

Para el acceso de datos se utiliza la API REST para realizar las peticiones al servidor y este devuelva una respuesta con los datos para representarlos en los marcadores.

En la barra de acciones (menú superior de la pantalla) tenemos como primera opción navegar a otras actividades, también un buscador para localizar lugares del mapa y actualizar el contenido de la pantalla (los datos de los marcadores).

Igual que en otras actividades pulsando el botón atrás, el usuario puede salir de la aplicación seleccionando la opción afirmativa del cuadro de diálogo.

Activity: ListIncidenceActivity.java

Actividad donde el usuario puede visualizar todas las incidencias en tiempo real. Para llevar a cabo la tarea se ha utilizado la RecyclerView que permite mostrar grandes colecciones o conjuntos de datos.

Dichos datos se muestran utilizando un adaptador, en este caso se usa la clase IncidenceAdapter.java. Se pude decir, que los adaptadores son colecciones de datos, que asignamos a una vista (RecylcerView) para que esta los muestre. El adaptador tiene su propio layout para distribuir como se van a mostrar los datos.

Para la obtención de los datos se utiliza el mismo procedimiento que en la actividad TrafficActivity.java.

En la barra de acciones tenemos la opción de navegar a otras actividades y actualizar el contenido de la pantalla.

Igual que en otras actividades pulsando el botón atrás, el usuario puede salir de la aplicación seleccionando la opción afirmativa del cuadro de diálogo.

Activity: ListRadarActivity

Actividad donde el usuario puede visualizar todos los radares en tiempo real de Cataluña. Esta actividad se ha implementado exactamente igual que ListIncidenceActivity.java, la única diferencia es el contenido de la información del adaptador que necesita la actividad (RadarAdapter.java).

En la barra de acciones tenemos la opción de navegar a otras actividades y actualizar el contenido de la pantalla.

Igual que en otras actividades pulsando el botón atrás, el usuario puede salir de la aplicación seleccionando la opción afirmativa del cuadro de diálogo.

Activity: ListCameraActivity

Actividad donde el usuario puede visualizar todas las cámaras en tiempo real de Cataluña. Esta actividad se ha implementado exactamente igual que ListIncidenceActivity.java, la única diferencia es el contenido de la información del adaptador que necesita la actividad (CameraAdapter.java).

En la barra de acciones tenemos la opción de navegar a otras actividades y actualizar el contenido de la pantalla.

Igual que en otras actividades pulsando el botón atrás, el usuario puede salir de la aplicación seleccionando la opción afirmativa del cuadro de diálogo.

Activity: NewsActivity

En esta actividad el usuario accederá a la página oficial del transit.gencat.cat, en la sección de noticias. Para mostrar la actividad se utiliza una vista que muestra páginas web (WebView). Es importante que en el AndroidManifest.xml se añada el permiso de acceso a Internet.

Debido a problemas de terceros, esta actividad no está disponible en castellano. Por ello se muestra el correspondiente mensaje para que el usuario sepa que no es un error de la aplicación.

En la barra de acciones tenemos la opción de navegar a otras actividades y actualizar el contenido de la pantalla.

Igual que en otras actividades pulsando el botón atrás, el usuario puede salir de la aplicación seleccionando la opción afirmativa del cuadro de diálogo.

Activity: SettingsActivity

En ella el usuario puede configurar sus preferencias, como cambiar el idioma y mostrar u ocultar los marcadores en el mapa. Los cambios efectuados se persisten en el teléfono.

Para el cambio de idioma, se le notifica a la aplicación que carpeta debe acceder para cargar los textos del idioma correspondiente. En el apartado “Carpeta res” se detalla más.

En el caso de mostrar elementos guarda una variable de tipo booleano (verdadero o falso).

En la barra de acciones solo hay la opción de navegar a otras actividades.

Igual que en otras actividades pulsando el botón atrás, el usuario puede salir de la aplicación seleccionando la opción afirmativa del cuadro de diálogo.

3.3.2. Carpeta utils

La carpeta "utils" tiene clases auxiliares que ayudarán a resolver ciertas operaciones que deba realizar la aplicación. En la imagen inferior se muestra en el proyecto la carpeta y sus entidades.

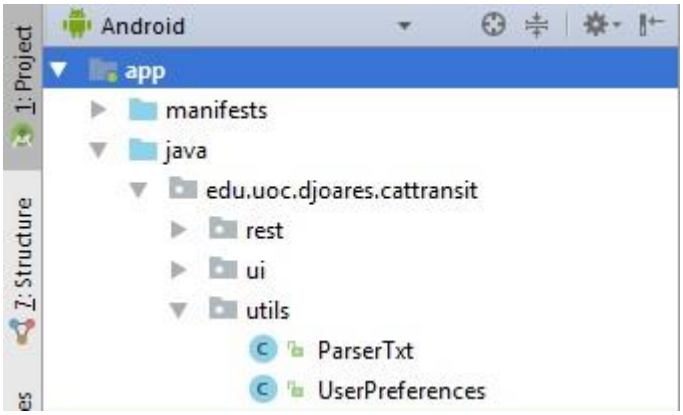

Ilustración 16: Detalle de la carpeta utils

La clase "ParserTxt" tiene la función de convertir los datos de los radares recibidos por la REST en objetos java. El documento en formato texto contiene cinco datos estructurados en cinco columnas tabuladas. Los datos almacenados son el nombre de la carretera, el punto kilométrico, velocidad máxima y sus coordenadas UTM.

Ejemplo:

```
Via     PK       Velocitat    X             Y
A-2     445,35   120          288075,4643   4601625,063
A-2     477,4    120          315966,2147   4611784,9372
A-2     510      120          347348,6149   4613312,8914
```

La anchura de cada columna es igual para cada atributo (12 caracteres), lo que facilitó el poder capturar los datos y transformarlo posteriormente en un objeto. Sin embargo, las coordenadas UTM no sirven para representar el radar en el mapa de Google, ya que este utiliza grados decimales. Para la conversión se utilizó la librería JCoord y así obtener la latitud y longitud de cada radar.

La clase "UserPreferences" utiliza la clase SharedPreferences para almacenar las preferencias del usuario. Las preferencias no son más que datos que una aplicación debe guardar para personalizar la experiencia del usuario, en la aplicación se guardará el idioma preferente del usuario y si quiere mostrar los elementos (radares, incidencias y cámaras) en el mapa. La clase tiene dos funciones para cada una ella de las preferencias, guardarla en la memoria del dispositivo y comprobar cuál es.

3.4. Carpeta res

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, layouts, cadenas de texto, dimensiones, etc. Los diferentes tipos de recursos se distribuyen en las siguientes subcarpetas que se detallan a continuación.

Carpeta drawable: Contiene las imágenes y otros elementos gráficos usados por la aplicación.

Carpeta mipmap: Contiene los iconos de lanzamiento de la aplicación (el icono aparecerá en el menú de aplicaciones del dispositivo) para las distintas densidades de pantalla existentes.

Carpeta layout: contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica, además de otros elementos como la barra de acción o los adaptadores para mostrar la información en los listados.

Se ha creado un layout para cada pantalla y adaptador además de la barra de acción.

Carpeta menu: Contiene la definición XML de los menús de la aplicación.

Carpeta values: Contiene ficheros XML de recursos de la aplicación. Estos son los que se incluyen la carpeta:

-colors: define los valores de los colores utilizados en la aplicación.

-dimens: define las dimensiones, tanto alturas como tamaños de fuentes de la interfaz de usuario.

-strings: define cadenas de texto usadas en la aplicación. Por ejemplo, colocar los títulos de las ventanas. En el proyecto se ha definido dos ficheros strings, uno para castellano y otro para el catalán. Estos se guardan en las carpetas values-es y values-ca respectivamente.

-styles: define los estilos usados en la aplicación. Pueden ser aplicados a los elementos de la interfaz de usuario, de modo que separamos la plantilla de las funcionalidades.

-google-maps-api: define la clave API para acceder a los servidores de Google Maps.

3.5. Google API v3

Para las aplicaciones de movilidad el uso de mapas es fundamental. Nos permite visualizar en el dispositivo un mapa donde se puede localizar un punto definido, en el proyecto los elementos a mostrar como incidencias. Además, visualmente es atractivo de cara al usuario.

Los dispositivos Android se integran correctamente con la aplicación GoogleMaps, obviamente porque son de la misma empresa. Por eso se ha decidido integrar la última versión disponible de GoogleMaps v3. Los requisitos para su uso son una versión del JDK 7.0 o superior y Android Studio 2.0.

Para un correcto funcionamiento en la aplicación Android, se han seguido las siguientes tareas que se detallan a continuación.

Primero se descargan las librerías necesarias de Google Play Services e integrarlas en nuestro proyecto en Android Studio.

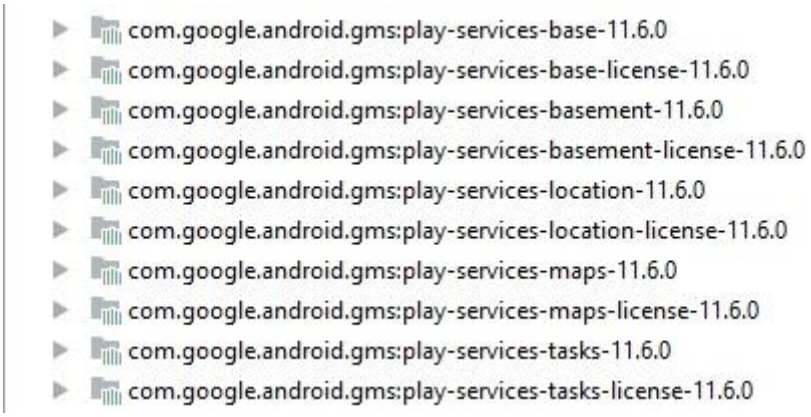

Ilustración 17: Librerías Google-Service

Tenemos que crear un proyecto CatTrànsit en Google, esto nos permitirá generar un API KEY que posteriormente debemos declara en el fichero AndroidManifest.xml.

En el menú Panel de control podemos seleccionar que servicios queremos integrar en el proyecto, en nuestro caso seleccionamos Google Maps Android API.

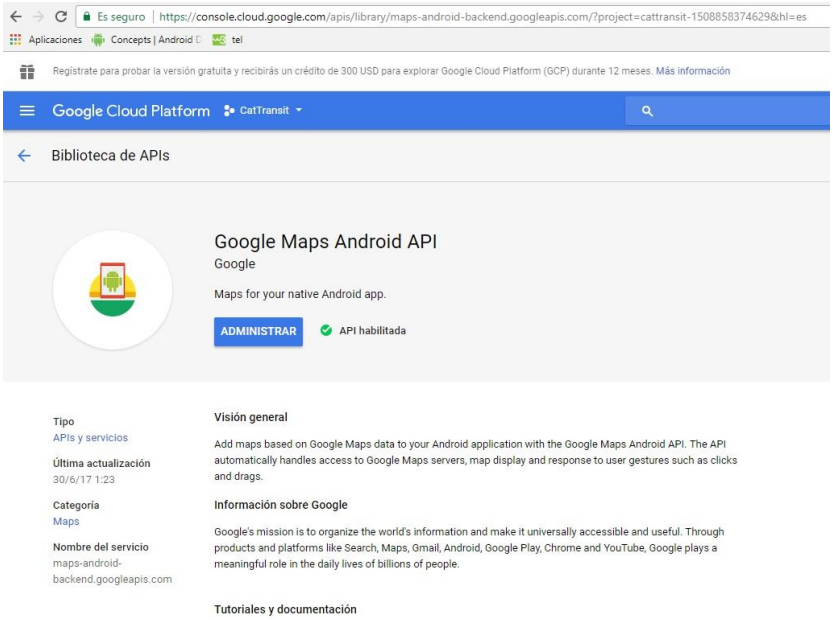

Ilustración 18: Pantalla para habilitar la API Google Maps

En la página Credenciales obtenemos la clave de API: En el cuadro de diálogo debemos introducir el nombre del paquete y la huella digital SHA-1 que se encuentra en el archivo google_maps_api.xml. Este archivo se genera cuando se crea una activity de GoogleMaps. En este mismo archivo introduciremos también la clave API obtenida.

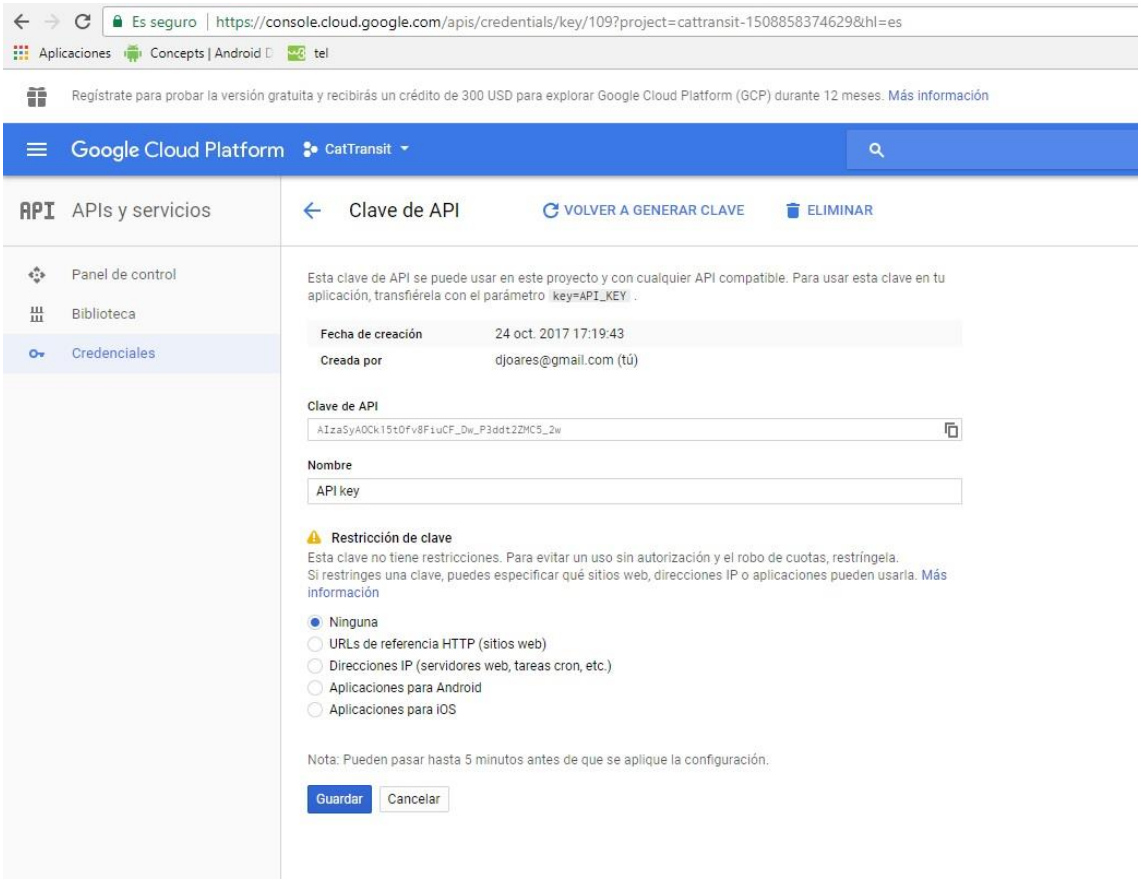

Ilustración 19: Pantalla de credenciales de la API Google Maps

En este punto tenemos el entorno preparado para trabajar con las librerías de la API de Google Maps.

Una de las funciones que dispone es localizar la posición del dispositivo o añadir marcas en el mapa introduciendo en ellas la información que se necesite. Para más información de las funciones que se han utilizado, ver el archivo de código fuente del proyecto.

3.6. Retrofit

Una REST es una interfaz entre sistemas que sirven para obtener datos o realizar operaciones sobre ellos en todos los formatos posibles como JSON o XML.

Las operaciones más importantes que pueden realizar con los datos son GET, POST, PUT, y DELETE (consultar, crear, editar y eliminar).

La ventaja de utilizar REST es que separa la interfaz de usuario del servidor y el almacenamiento de datos.

Que el cliente y el servidor estén separados la ventaja de que el producto se puede escalar sin muchos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte las aplicaciones en productos más flexibles.

La REST es totalmente independiente del tipo de lenguaje o plataforma. Lo importante es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio usado, en este caso XML y TXT.

Para crear la REST de la aplicación se ha utilizado la librería Retrofit. Las partes que lo componen son el modelo de cada entidad, la base del adaptador, las constantes y el servicio.

Para implementar la REST en el proyecto se decidió usar esta API porque es simple y fácil de aprender. También porque se utilizó durante el máster y eso es una ayuda extra.

Para poder usar Retrofit en el proyecto se añade la dependencia vía Build.gradle(Module:app). Se añade el conversor de XML que se usará para obtener los objetos mediante lenguaje XML.

```
compile 'com.squareup.retrofit2:retrofit:2.3.0'
compile('com.squareup.retrofit2:converter-simplexml:2.3.0')
{
    exclude module: 'stax-api'
    exclude module: 'stax'
    exclude module: 'xpp3'
}
```

También se añade la siguiente dependencia para poder visualizar las peticiones en una lista.

```
compile 'com.android.support:recyclerview-v7:25.3.1'
```

Por último, es necesaria la conexión a Internet, por lo que se añade el permiso en el archivo AndroidManifest.xml.

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

Ahora el proyecto está preparado para utilizar la API Retrofit.

4. Mejoras de versiones futuras

En este punto se explicará algunas nuevas funcionalidades que no se han podido desarrollar por falta de tiempo o por otras circunstancias.

Una de las mejoras, es que el usuario pueda seleccionar que elementos en concreto quiera visualizar en el mapa. Esto se implementaría en la leyenda del mapa y la opción de mostrar elementos de la pantalla configuración desaparecería.

En la actividad del tráfico se puede observar que las imágenes de los marcadores tardan en cargarse, lo cual, sería conveniente mostrar un barra de carga para informar al usuario que debe esperar para visualizar la imagen. De hecho esta implementado de esta manera en el adaptador, pero por desconocimiento, en las ventanas de los marcadores no se pueden incluir animaciones. Por falta de tiempo, no se ha podido implementar esta parte de otra manera distinta a la que se había planteado inicialmente pero en futuras versiones se corregirá.

Otra opción a añadir es agrupar los elementos de los listados por zonas o comarcas, permitiendo al usuario encontrar incidencias u otra información de manera más rápida.

Por último, en la actividad de noticias, se informa al usuario que esta funcionalidad no está disponible en castellano por motivos de terceros. Se ha puesto en contacto con los responsables de la información para determinar si se traducirá la página web al castellano y estamos a la espera de una respuesta.

5. Conclusiones

Realizar el TFM se ha podido aprender muchos conocimientos nuevos así como ampliar otros. La elección de realizar el desarrollo de una aplicación en Android es para ampliar los conocimientos en la creación de aplicaciones para dicha plataforma

La elección de tecnologías ha sido acertada. Android Studio es el IDE oficial para desarrollo de aplicaciones para la misma plataforma. En el caso de las librerías Retrofit y GoogleMaps, la primera es una herramienta potente y fácil de aprender y la segunda, muy útil para el proyecto. En ambas se tenían conocimientos previos que se han podido ampliar aprendiendo nuevas funcionalidades.

Por otro lado, se ha tenido la oportunidad de desempeñar funciones que no se habían hecho antes como realizar el requisito de funcionalidades o la planificación, y fue una dificultad añadida. Al realizar dichas fases, se ha podido ver la importancia que estas fases pueden tener para poder encarar bien el desarrollo de la aplicación de forma correcta y que en el principio del proyecto no se les dio la importancia que merecen. En general, decir que el aprendizaje de nuevas funcionalidades era la esperada y no se ha podido cumplir uno de los objetivos porque la funcionalidad a implementar no era importante y por falta de tiempo.

Para finalizar, el TFM ha sido una gran experiencia para poder conocer todas las fases de desarrollo de una aplicación móvil.

6. Glosario

- **Activity:** componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción.
- **Android:** Sistema operativo basado en el núcleo Linux. Desarrollado inicialmente por Android Inc., y que posteriormente Google compro.
- **API:** Application Programming Interface, son un conjunto de funciones que permiten a los desarrolladores crear aplicaciones simplificando el trabajo de los creadores de programas.
- **APK:** aplicación Android en forma de archivo instalable.
- **Build.gradle:** contiene la información necesaria para la compilación del proyecto.
- **Clase:** abstracción que define un tipo de objeto especificando qué propiedades y operaciones disponibles va a tener.
- **Cluster:** conjunto o cúmulo de elementos.
- **Compilar:** Proceso de traducción de un código fuente (escrito en lenguaje de programación de alto nivel) a lenguaje máquina.
- **Coordenadas UTM:** Universal Transversal de Marcator, sistema basado en la proyección cartográfica de Mercator, sus unidades son lo metros a nivel del mar, que es la base del sistema de referencia.
- **Firebase:** plataforma de desarrollo móvil en la nube de Google.
- **Google Maps:** servidor de aplicaciones de mapas que ofrece fotografías por satélite, mostrar rutas entre ubicaciones o imágenes de calles.
- **GPS:** Global Positioning System, sistema que permite determinar la posición de un objeto respecto la Tierra.
- **IDE:** aplicación informática que proporciona un conjunto de herramientas para los desarrolladores.
- **iOS:** Sistema operativo desarrollado por Apple Inc., utilizado en dispositivos exclusivos de la compañía Apple.
- **JDK:** Java Development Kit, conjunto de herramientas de desarrollo para la creación de programas en Java.
- **Layout:** estructura visual para una interfaz de usuario, es decir, aquella que hace de intermediario entre el terminal móvil y el usuario.
- **Manifest:** archivo de configuración donde podemos aplicar las configuraciones básicas de nuestra aplicación.
- **MVC:** Modelo-Vista-Controlador es un patrón de arquitectura de software que separa la lógica de la aplicación de la vista.
- **Objetos java:** entidad existente en la memoria del ordenador que tiene unas propiedades, atributos y funciones.
- **OkHttp:** OkHttp es un proyecto de código abierto diseñado para ser un cliente HTTP eficiente.
- **Open Data:** son datos que pueden ser reutilizados y redistribuidos libremente por cualquier persona, siempre que no se alteren estos.
- **PAC:** Prueba de Evaluación Continua.

- **REST:** REpresentational State Transfer, servicio que permite añadir, eliminar, consultar o editar diferentes productos utilizando una serie de métodos.
- **Retrofit:** API REST desarrollada por Square para optimizar el envío de peticiones Http en aplicaciones Android hacia servidores externos.
- **SCT:** Servei Català de trànsit.
- **SDK:** conjunto de herramientas de desarrollo de software.
- **Serializar:** proceso de codificación de un objeto en un medio de almacenamiento con el fin de transmitirlo a través de una conexión en red en un formato más legible como XML.
- **TFM:** Trabajo Final de Master.
- **Txt:** extensión de documentos de texto plano.
- **URL:** Uniform Resource Locator, secuencia de caracteres que sigue un estándar y que permite denominar recursos dentro del entorno de Internet para que puedan ser localizados.
- **USB:** Universal Serial Bus, hace referencia a un protocolo de conexión que permite enlazar diversos periféricos a un dispositivo electrónico para el intercambio de datos.
- **Waterfall:** método de desarrollo de software que se rige por seguir de manera secuencial las diferentes etapas del proceso del desarrollo, una etapa no puede iniciarse si la anterior no se ha terminado.
- **XML:** siglas de eXtensible Markup Language, lenguaje desarrollado por el World Wide Web Consortium (W3C) que permite la organización y etiquetado de documentos.

7. Bibliografía

1. http://dadesobertes.gencat.cat/es/: Es el portal de la Generalitat de Cataluña que difunde datos abiertos para que ser reutilizados por terceros. (01/10/2017)
2. http://transit.gencat.cat/ca/inici/: Portal web del Servei Català de Trànsit. (04/10/2017)
3. http://transit.gencat.cat/web/.content/documents/seguretat_viaria/radars.txt: Página web que contiene los datos de los radares de Cataluña en formato texto. (05/10/2017)
4. http://www.gencat.cat/transit/opendata/cameres.xml: Página web que contiene los datos las cámaras viarias de Cataluña en formato XML.
5. http://www.gencat.cat/transit/opendata/incidenciesGML.xml: Página web que contiene los datos de las incidencias del tráfico de Cataluña en formato XML. (05/10/2017)
6. http://transit.gencat.cat/ca/el_servei/premsa_i_comunicacio/noticies/: Página web que contiene las noticias referentes al tráfico de Cataluña. (05/10/2017)
7. https://www.freepik.com/free-icons/multimedia: Web para descarga de iconos. (14/10/2017)
8. Usuarios y sistemas interactivos: Eva Patricia Gil, Eva de Lera Tatjer, Antònia Monjo Palau, documento pdf de la asignatura Diseño de productos interactivos multidispositivo. (19/10/2017)
9. https://developers.google.com/maps/documentation/android-api/start?hl=es-419: Página web oficial de Google para la instalación de la API Google Maps. (02/11/2017)
10. http://square.github.io/retrofit/: Página web oficial de Square para la instalación y uso de la librería Retrofit. (02/11/2017)
11. http://simple.sourceforge.net/: Página web para la serialización de objetos XML. (05/11/2017)
12. http://www.jstott.me.uk/jcoord/: Página web para la descarga de la librería Jcoord. (10/11/2017)
13. https://developers.google.com/maps/documentation/javascript/markers?hl=es-419: Página web oficial de Google relacionada con los marcadores de Google Maps. (23/11/2017)

8. Anexos

8.1. Instalación

Una vez descargado el archivo APK, para poder instalarlo en el sistema, este debe habilitar la instalación de aplicaciones que no vengan desde Google Play.

Para ello, hay que dirigirse al menú general->Seguridad. En este submenú aparecerá una opción llamada Orígenes desconocidos, la cual debemos activar. En el momento de activarlo, aparecerá una advertencia informando de los riesgos de instalar aplicaciones que no procedan de la tienda de Google Play. Hay que aceptarlos.

En el siguiente paso conectamos el dispositivo al PC vía USB y guardamos el APK en la carpeta Downloads del dispositivo.

Por último, en el dispositivo nos dirigimos al gestor de archivos y localizamos el APK. Pulsando en el archivo el sistema nos informará si requiere de algún permiso. Una vez aceptados estos permisos, la aplicación se instalará y estará disponible en el sistema para ser usada.

8.2. Manual de usuario

Una vez instalada la aplicación, la tendremos disponible pulsando el icono.

Ilustración 20: Icono de la aplicación

La primera pantalla que se visualiza en ejecutar la aplicación es la SplashActivity que pasado un segundo aparece el mapa de Google, la TrafficActivity.

TrafficActivity

Si el dispositivo tiene la versión Android 6.0 (marshmallow) o posterior, el usuario no necesita otorgar permisos cuando instala la aplicación. En caso contrario el sistema muestra al usuario un cuadro de diálogo en el que se solicita el permiso, debe seleccionar que si para poder utilizar el mapa.

En está pantalla podemos visualizar la densidad del tráfico definido en los siguientes colores, verde poca densidad, amarillo densidad moderada, rojo y negro mucha densidad y en el gris no hay datos disponibles.

Si la cantidad de elementos en el mapa están muy próximos estos se agrupan en clústeres. Realizando zoom en la pantalla se pueden ver estos y la información que contienen.

También cuenta con el botón de localización, al pulsarlo la cámara se desplazará hacia la localización del dispositivo.

Debajo de este está el botón información, que incluye una leyenda con el significado de cada icono en el mapa.

Por último en la barra de acción se muestran tres iconos que son de izquierda a derecha los siguientes:

-Menú: muestra el listado de actividades.

-Buscador: el usuario puede introducir un nombre de una localidad para que Google Maps la localice y la muestre.

-Refrescar: actualiza el contenido del mapa.

En otras actividades la barra de acción es idéntica salvo por el buscador que es exclusivo del mapa.

Ilustración 21: Pantalla de actividad de tráfico

IncidenceActivity

Muestra el listado de incidencias con su información más relevante en tiempo real.

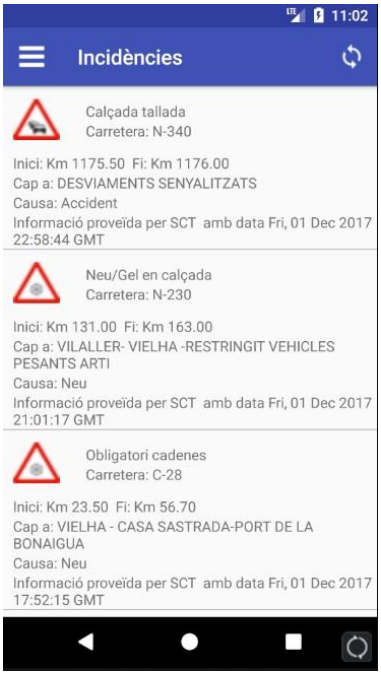

Ilustración 22: Pantalla de actividad de incidencias

NewsActivity

Muestra las noticias del portal transit.gencat.cat. En estos momentos por problemas ajenos, esta actividad solo estará disponible en catalán.

Ilustración 23: Pantalla de la actividad de noticias

CameraActivity

Muestra el listado de cámaras con su información más relevante en tiempo real.

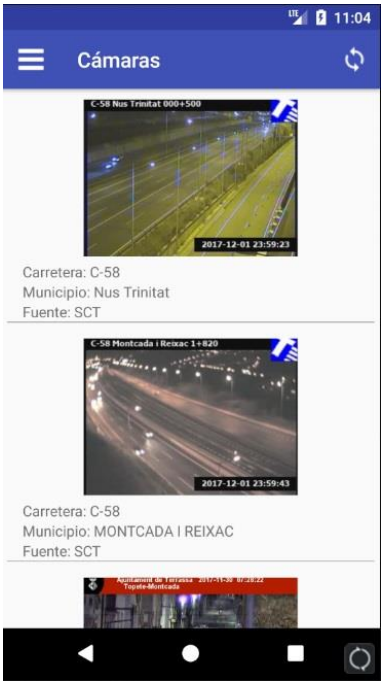

Ilustración 24: Pantalla de la actividad de cámaras

RadarActivity

Muestra el listado de radares con su información más relevante en tiempo real.

Ilustración 25: Pantalla de la actividad de radares

SettingsActivity

El usuario puede definir en qué idioma quiere visualizar la aplicación y si quiere mostrar los elementos en la mapa de Google.

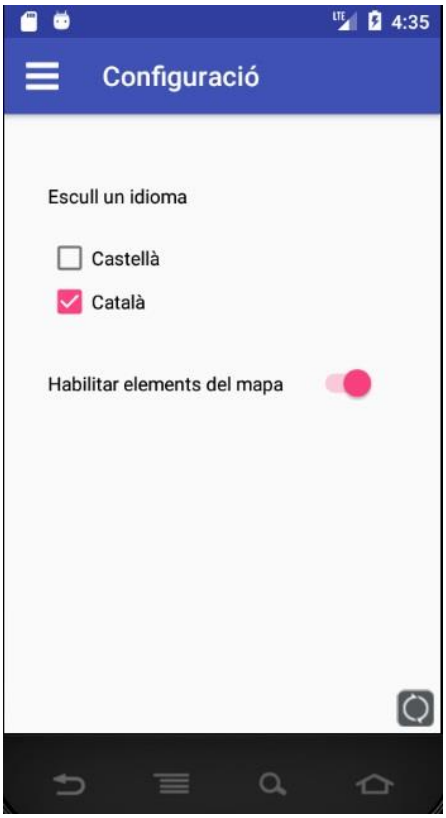

Ilustración 26: Pantalla de la actividad de configuración

Por último, para salir de la aplicación se pulsa el botón de retroceso (Back button) en la parte inferior del dispositivo. Se mostrará un mensaje para confirmar si quiere el usuario salir de la aplicación.