Aplicación cámaras de tráfico de Madrid

Desarrollo de aplicaciones móviles con Android

Contenido

Introducción

La aplicación que va a desarrollar deberá integrar la mayoría de los conceptos de programación en Android estudiados en esta asignatura. Además, tendrá que investigar por su cuenta otros conceptos para poder realizar la aplicación.

La aplicación que va a implementar está basada en la del portal informo (figura 1) que el ayuntamiento de Madrid pone a disposición del ciudadano para el seguimiento del tráfico en tiempo real. Para su realización se usará la información del *Portal de datos abiertos del Ayuntamiento de Madrid*, concretamente el fichero KML que puede obtenerse de esta página, o bien directamente del siguiente enlace: http://informo.madrid.es/informo/tmadrid/CCTV.kml.

Figura 1: Página del portal informo

Descripción básica

De forma resumida, la aplicación a desarrollar debe visualizar un listado de los nombres de las cámaras de tráfico, permitir al usuario seleccionar una de ellas, mostrar una foto en tiempo real de lo que se ve desde esa cámara, y al hacer clic en la foto, ver su posición en un mapa. Toda la información que necesitará la aplicación la debe obtener del documento KLM que debe descargar cada vez que se inicia esta.

La aplicación constará de dos actividades:

- La primera, llamada actividad principal en este documento, descargará el documento KML, lo procesará para extraer la información pertinente y finalmente mostrará en la pantalla del terminal una lista con los nombres de las cámaras. Al seleccionar una de ellas deberá descargar la foto que en ese momento ofrece la cámara y visualizarla, como puede verse en las figuras ¡Error! No se encuentra el origen de la referencia. y Figura 3

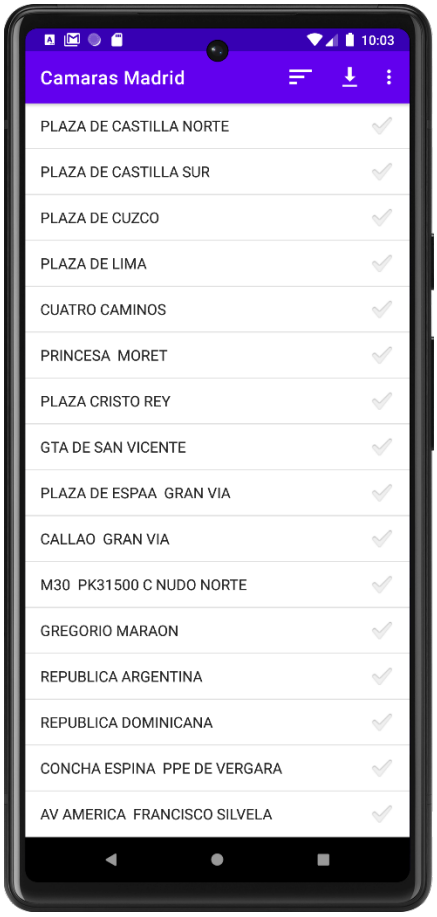


Figura 2

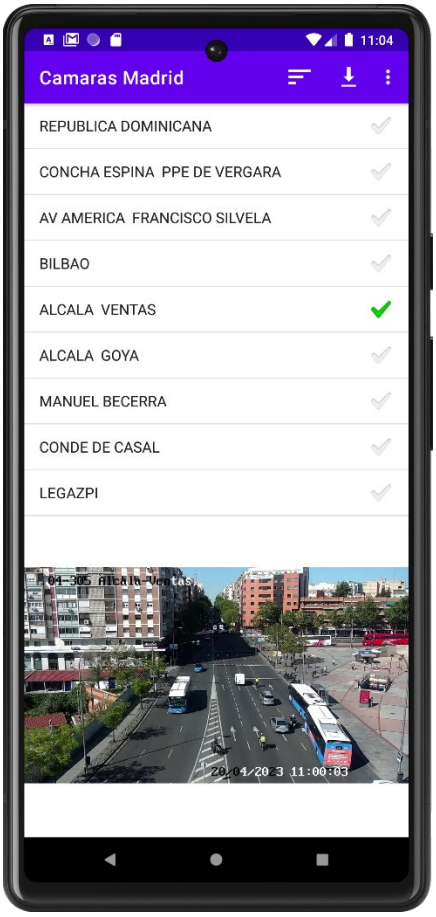


Figura 3

- La segunda actividad, llamada actividad del mapa en este documento, se abrirá al hacer clic sobre la imagen de la actividad principal y deberá mostrar un mapa con un marcador, situado en la posición geográfica de la cámara seleccionada en la actividad principal, con el nombre de la cámara. El mapa deberá poder visualizarse en los cuatro tipos de vistas que pueden verse en las figuras Figura 4 y Figura 5 (solo se muestran dos de ellos), gracias a los elementos de selección disponibles en la parte superior. La cámara seleccionada deberá mostrarse en un color que sea diferente al rojo de por defecto de los marcadores. Por ejemplo, en azul.

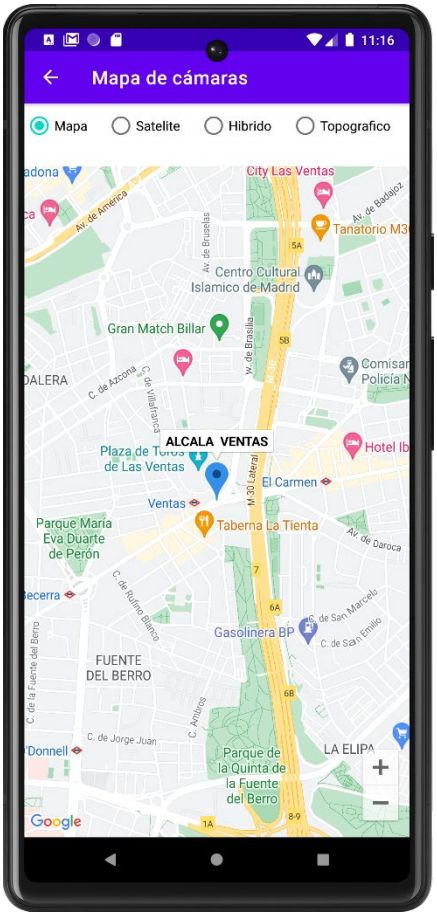


Figura 4

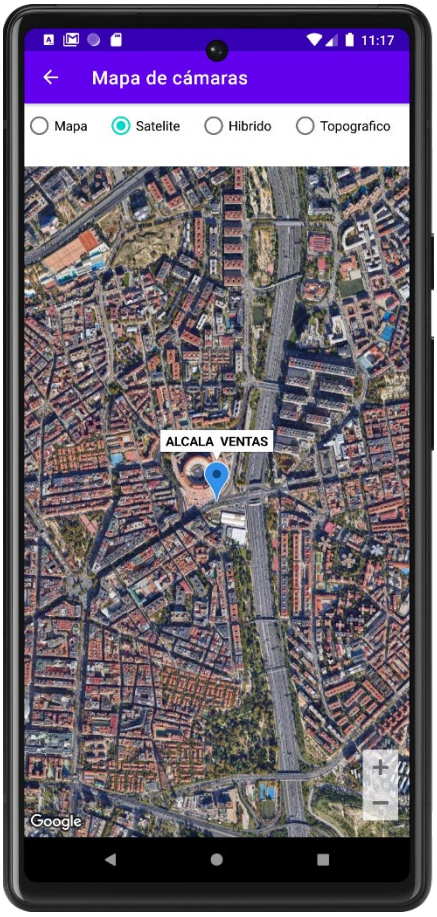


Figura 5

Especificaciones

A continuación, se describen las funcionalidades que deben añadirse a la descripción básica comentada en el apartado anterior. No es necesario que el diseño de cada funcionalidad se realice como se describe en este apartado o se ve en las capturas de pantalla, se debe implementar la funcionalidad, pero el diseño puede ser diferente. Por ejemplo, si en el apartado anterior se dijo que para abrir la actividad del mapa se debe hacer clic en la imagen de una cámara, también se podría implementar esta funcionalidad mediante un botón que salga junto a la imagen de la cámara. y que tenga que hacerse clic en ese botón.

Indicación de progreso

Durante la carga y procesamiento del fichero *KML*, así como durante la carga de la imagen, deberá mostrase una indicación de progreso.

En las figuras 6, 7 y 8 se muestra un ejemplo de cómo podría indicarse el progreso de la descarga del fichero *KLM* (figura 6), el análisis de este (figura 7), o de la imagen (figura 8), mediante una barra de progreso horizontal, o mediante una barra de progreso circular. Puesto que para la descarga del fichero KLM, el servidor informa del tamaño del recurso, la barra de progreso horizontal tiene que informar del progreso real, avanzando con las líneas obtenidas del servidor.

Figura 6. Espera mientras descarga el fichero KLM de internet

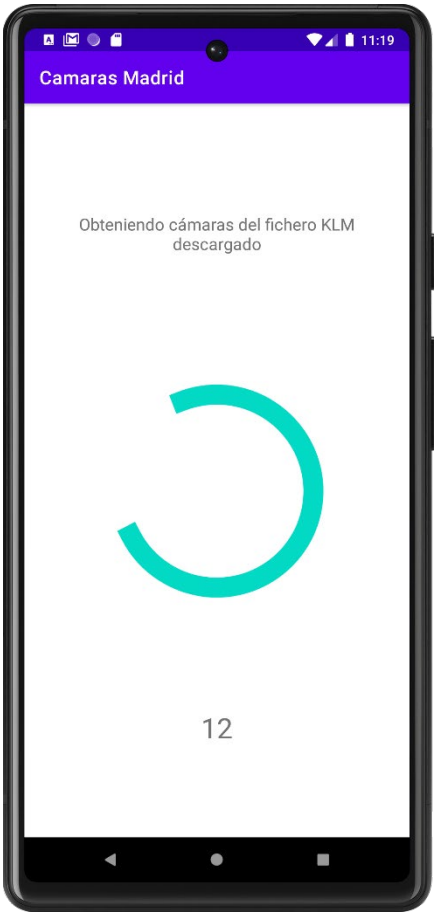


Figura 7. Espera mientras procesa el fichero KLM descargado

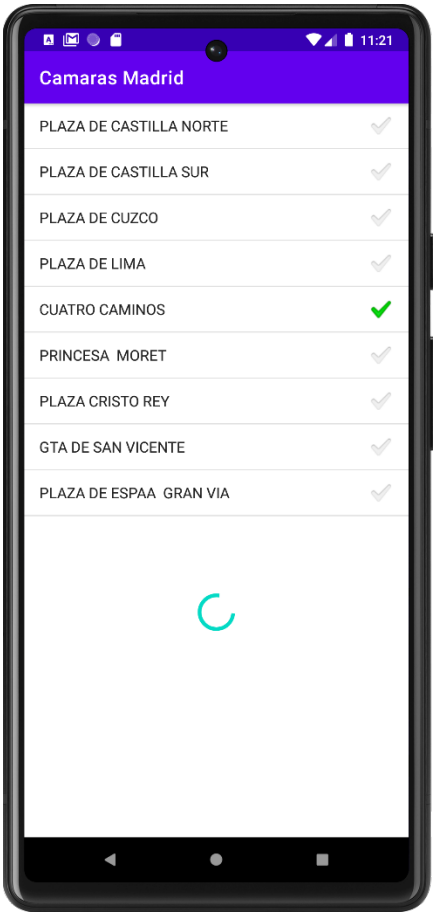


Figura 8. Espera mientras descarga la foto de la cámara de internet

Cambio de orientación

La aplicación deberá controlar el cambio de orientación del terminal. En este caso, cuando la orientación del terminal se corresponda con el modo horizontal, la actividad principal se visualizará tal y como se muestra en la Figura 9.

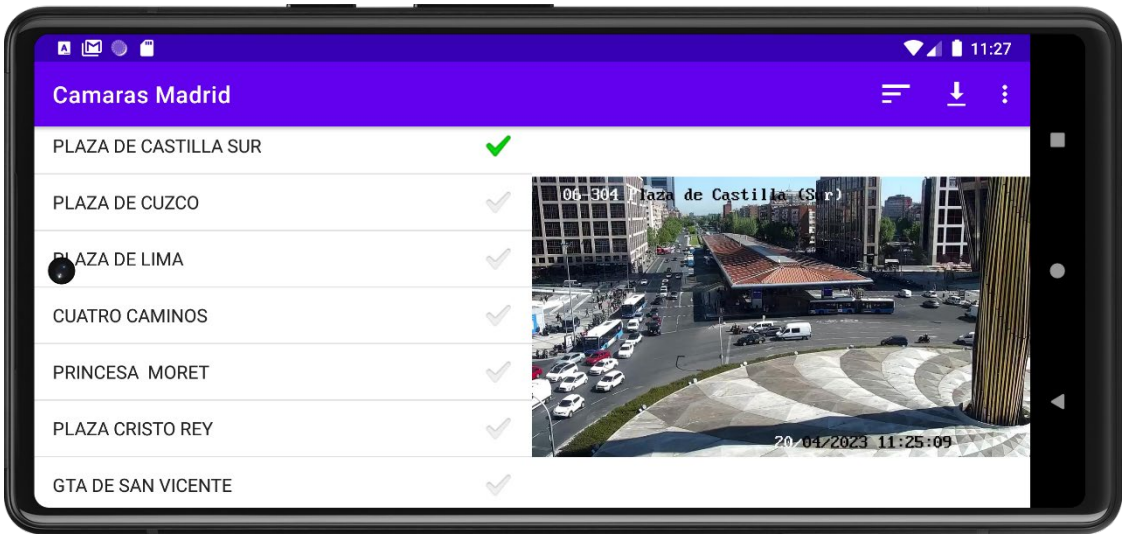


Figura 9

La actividad del mapa no necesita un diseño específico para el modo horizontal, pudiendo usarse el que tiene por defecto para el modo vertical.

Carga del fichero *KLM* y procesamiento

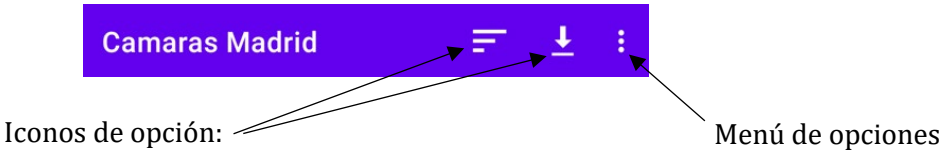
Cuando la aplicación arranca por primera vez tendrá que descargar de internet el fichero *KLM* y almacenarlo en el sistema de almacenamiento del terminal. A continuación, deberá procesarlo para guardar en memoria el listado de cámaras junto con los datos de cada una: nombre, ubicación y url (para obtener la foto que ofrece la cámara).

Una vez realizada esta operación anotará en un fichero *SharedPreferences* la fecha en la que ha se ha descargado el fichero *KLM*, de forma que, en otra ejecución de la aplicación, no sea necesario descargar de nuevo el fichero *KLM*. Concretamente lo que debe hacer la aplicación, cada vez que arranque, es obtener la fecha guardada en el fichero *SharedPreferences* y entonces preguntar al usuario, mediante un cuadro de diálogo, si quiere descargar de nuevo el fichero KLM de internet o recuperarlo del sistema de almacenamiento. En el cuadro de diálogo deberá informar de la fecha y hora en la que guardó el fichero KLM.

Se deberá tener en cuenta que, si es la primera vez que arranca la aplicación y por tanto nunca se ha descargado el fichero de internet, o bien el usuario ha decidido borrar los datos de la aplicación, no existirá nada guardado en *SharedPreferences*, por lo que no se deberá mostrar el cuadro de diálogo, sino que habrá que descargar directamente el fichero de internet.

Barra de herramientas

La actividad principal debe disponer de una barra de herramientas con dos iconos y un menú de opciones a la derecha. Gracias a estos elementos se podrán implementar las funcionalidades descritas en los siguientes apartados. En la siguiente figura puede verse un ejemplo de los iconos de opción, que no deben ser obligatoriamente los de esta figura:



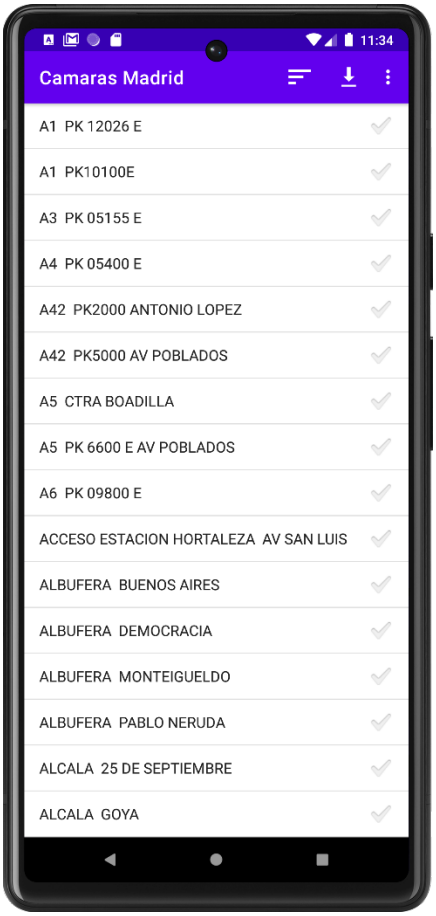
La actividad del mapa debe tener en la barra de título el icono de la flecha de vuelta atrás que le llevará a la actividad principal en el estado en el que se encontraba, sin recargar de nuevo el listado de cámaras:

Iconos de vuelta atrás

Opción ordenar

A esta opción se accederá mediante un icono de la barra de herramientas de la actividad principal.

Al elegir esta opción el listado de cámaras se mostrará por orden alfabético del nombre de las cámaras, como puede verse en la siguiente figura, sin necesidad de descargarlo de nuevo de internet, o leerlo de nuevo del almacenamiento:



Al pulsarlo de nuevo el listado de cámaras se ordenará a la inversa (de la Z a la A). De forma que cada vez que se pulse cambiará la ordenación.

Opción recarga de datos

A esta opción se accederá mediante un icono de la barra de herramientas de la actividad principal.

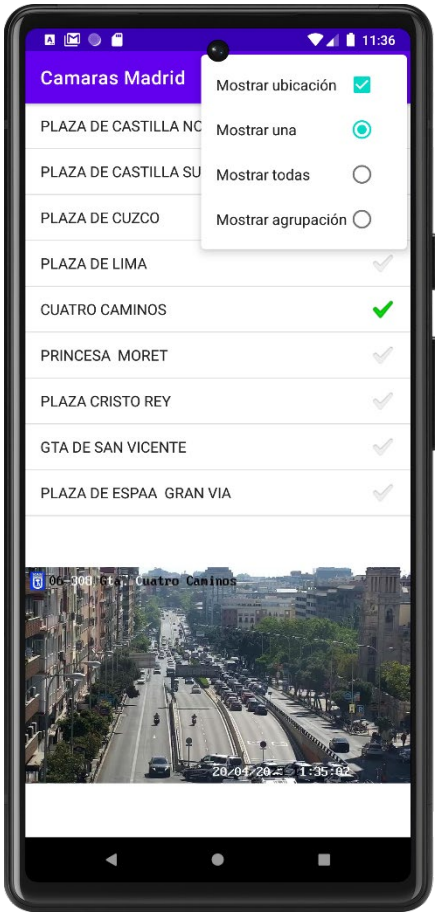
Al elegir esta opción se descargará de internet de nuevo el fichero *KLM*, se guardará en el sistema de almacenamiento interno y se procesará, anotando la fecha actual en el fichero *SharedPreferences*.

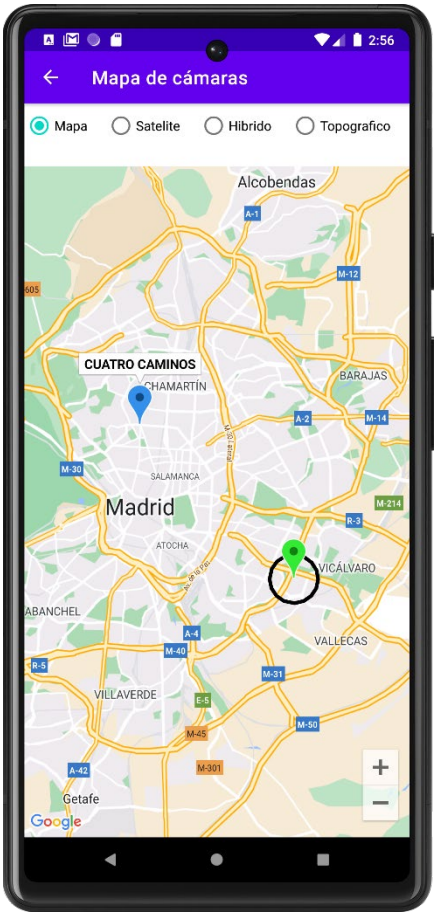
Opción mostrar la ubicación actual del terminal

A esta opción se accederá mediante una entrada del menú de opciones de la barra de herramientas de la actividad principal y será del tipo casilla de verificación (*checkbox*) para poderla habilitar y deshabilitar.

Cuando esta opción está habilitada, y el usuario pulse en la imagen de la cámara de la actividad principal, que previamente ha seleccionado del listado, se mostrará en la actividad del mapa con dos marcadores: el de la cámara seleccionada y el de la ubicación del terminal.

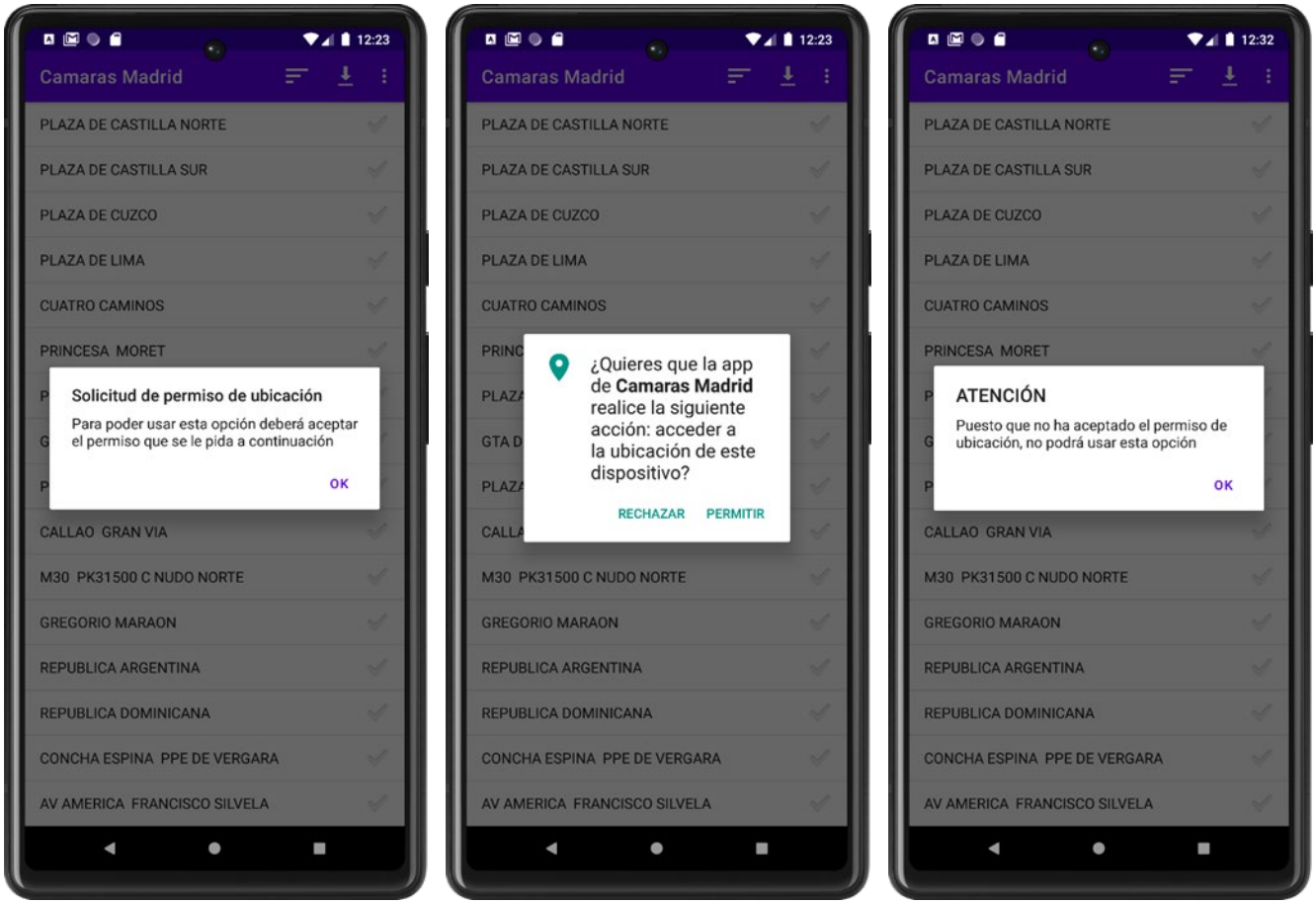
El marcador de la ubicación del terminal se deberá mostrar con un color diferente al de los marcadores de cámaras, por ejemplo en verde, y además se visualizará un círculo de 1km de radio centrado en el marcador, como puede verse en las siguientes figuras:





Para poder implementar esta opción se deberá usar el servicio de localización de Android que permite obtener la ubicación actual del terminal.

Puesto que para obtener la localización la aplicación necesita disponer de un permiso que debe ser concedido por parte del usuario, este permiso se debe solicitar cuando se acceda a esta opción en el caso de que no hubiera sido concedido anteriormente. Cuando se solicite el permiso tendrá que mostrarse un cuadro de diálogo que explique el motivo de la solicitud. Si el usuario deniega el permiso, se le informará mediante un cuadro de diálogo de que no podrá usar esta opción, no quedará marcada la casilla de verificación de la opción y por tanto no realizará su función. Si el permiso ha sido concedido, se quedará marcada la casilla de verificación de esta opción del menú. En las siguientes figuras tiene un ejemplo de lo que se podría ir mostrando durante el proceso de petición del permiso suponiendo que el usuario lo deniega:



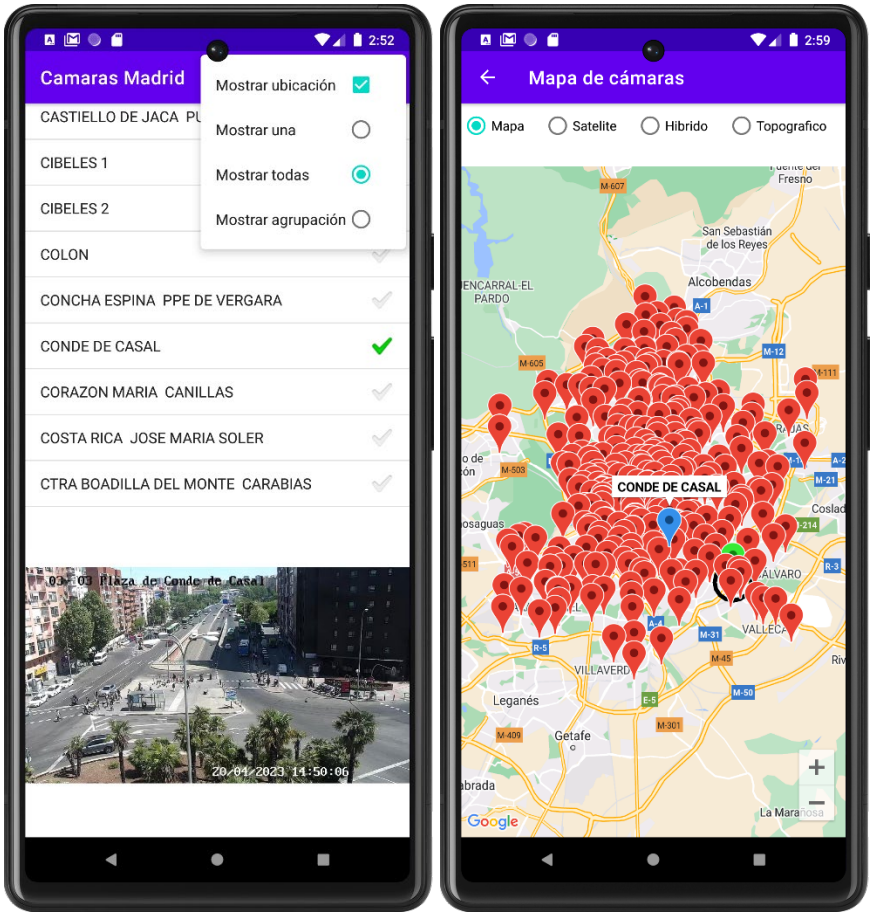
Opción mostrar todas las cámaras

A esta opción se accederá mediante una entrada del menú de opciones de la barra de herramientas de la actividad principal y será del tipo botón de opción (*radio button*) para poderla habilitar y deshabilitar de forma excluyente con las otras opciones de este tipo.

Cuando esta opción esté habilitada, en la actividad del mapa se visualizarán los marcadores de todas las cámaras, mostrando el nombre del marcador en la cámara que se ha seleccionado en la actividad principal.

Si estuviera habilitada la opción de ubicación actual del terminal, se añadirá un marcador con su localización y un círculo a su alrededor.

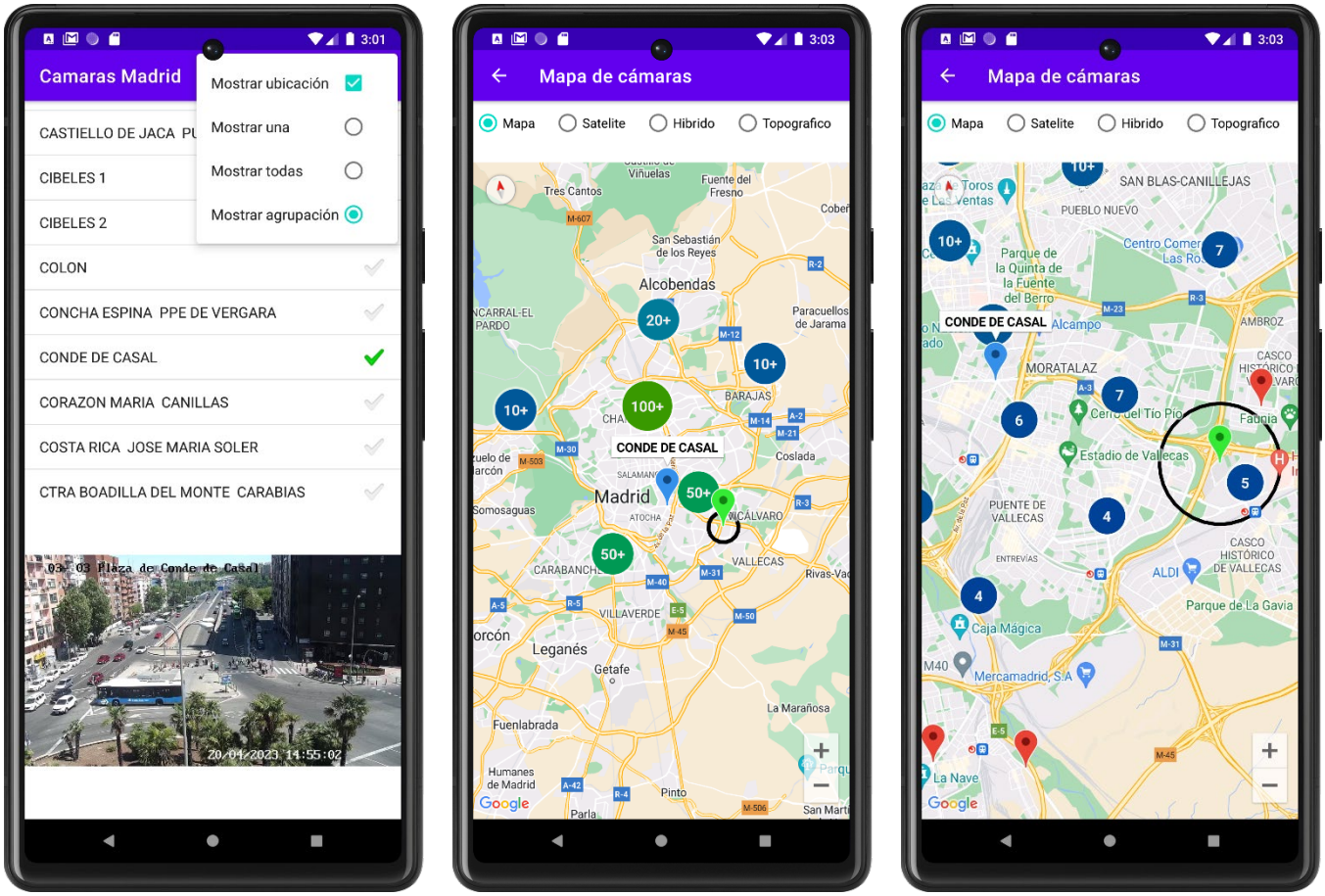
Los marcadores de todas las cámaras, excepto la seleccionada, tendrán el color rojo que se usa por defecto para los marcadores.



Opción mostrar agrupación de cámaras

A esta opción se accederá mediante una entrada del menú de opciones de la barra de herramientas de la actividad principal y será del tipo botón de opción (*radio button*) para poderla habilitar y deshabilitar de forma excluyente con las otras opciones de este tipo.

Como el número de marcadores es muy alto, con esta opción se usará la agrupación de marcadores para que el mapa no se vea tan cargado. Al realizar zoom en un área, algunas agrupaciones de marcadores desaparecerán para mostrar los marcadores que había en esa agrupación, permitiendo ver con más detalle los marcadores que hay en el área donde se ha hecho zoom. Al igual que con la opción de *mostrar todas*, el marcador de la cámara seleccionada en la actividad principal deberá mostrar el nombre de la cámara, y si está habilitada la opción de ubicación actual, se deberá mostrar un marcador en esa localización y un círculo a su alrededor.



Apariencia

La aplicación debe tener una apariencia en cuanto a colores, fondo, tipos de letras y tamaño, que sea diferente al tema por defecto que ofrece Android Studio al crear un nuevo proyecto. Para ello se debe definir un nuevo tema con nuevos estilos y colores a los de por defecto.

Icono de aplicación

La aplicación debe disponer de un icono a elección del estudiante.

Ampliaciones

Se podrán implementar otras funcionalidades no descritas en las especificaciones básicas para dotar de más funcionalidades a la aplicación. Algunos ejemplos son:

En la actividad principal:

- Usar un *ListView* personalizado con un diseño diferente a los que proporciona Android.
- Añadir un buscador de cámaras interactivo, de forma que al ir escribiendo el nombre se vaya filtrando las cámaras del listado.
- Dar la opción de mostrar la imagen de una cámara seleccionada ocupando toda la pantalla, auto-refrescándose cada cierto tiempo. Esta funcionalidad le permitiría a un usuario tener visible en tiempo real la foto que envía la cámara.
- Permitir al usuario guardar cámaras favoritas, seleccionándolas por ejemplo con un clic largo, de forma que estas cámaras se muestren siempre al principio del listado. Se sacarían de cámaras

favoritas con otro clic largo sobre su nombre. Estas cámaras favoritas podrían guardarse en *SharedPreferences* para futuras ejecuciones de la aplicación.
- Configurar la aplicación con el modo oscuro, bien mediante una opción del menú de la aplicación o con la configuración del sistema.

En la actividad del mapa:

- Cuando se haya seleccionado alguna de las opciones de mostrar la ubicación del terminal, o la de todas las cámaras, mostrar en la barra de herramientas un icono que al pulsarlo se haga un zoom a nivel de calles a la cámara seleccionada en la actividad anterior.
- Al hacer clic en el marcador de una cámara, mostrar en su información la imagen de la cámara.
- Si hay cámaras favoritas, cuando se muestren varias cámaras que estas tengan un marcador de color diferente y/o que permita mostrar solo las cámaras favoritas mediante una nueva opción en el menú de la actividad principal.
- Al pinchar en una cámara, implementar la funcionalidad de "cómo llegar" de forma que se cree una ruta desde la posición actual hasta esa cámara, con la posibilidad de abrir una nueva ventana que muestre una foto de todas las cámaras que hay en esa ruta para que así el usuario pueda ver el tráfico hasta su destino.

Restricciones de diseño

- La descarga del fichero `klm`, el análisis de su contenido para obtener la lista de cámaras y la carga de una imagen, debe poder hacerse simulando un pequeño retardo para que se puedan ver las barras de progreso. Ese retardo, en milisegundos, debe poder establecerse en un fichero `json` guardado en el directorio `Asset` con el nombre `configuración.json`, de manera que, sin modificar el código, el usuario pueda modificar el contenido de ese fichero para establecer el retardo que crea conveniente. El contenido de este fichero de configuración, que también se podría usar para otros propósitos, debe tener, al menos, un objeto JSON llamado "retardos" con este contenido:

```
{
  "retardos": {
    "descarga-klm": 5,
    "analisis": 30,
    "imagen": 50,
  }
}
```

- La actividad principal debe mostrar el progreso de la descarga y análisis del fichero KLM, además debe contener un fragmento estático para mostrar el listado de cámaras. La foto de la cámara seleccionada debe mostrarse en un fragmento dinámico.
- Cuando se muestre un solo marcador en el mapa, se debe usar un zoom a nivel de calles.
- En el mapa deben mostrarse los botones de zoom.
- Cuando se muestren varios marcadores en el mapa (tanto si se muestra el marcador de la cámara seleccionada y el del terminal, como cuando se muestran todas las cámaras), se debe calcular el zoom apropiado para que puedan verse todos los marcadores con un pequeño margen alrededor. Debe ser un zoom calculado por la aplicación, no un zoom impuesto por el código.
- Al cambiar la orientación, estando en la actividad principal, la cámara que estuviera seleccionada y su imagen debe permanecer seleccionada y mostrada la imagen. Igualmente, las opciones del menú que estuvieran habilitadas deberán seguir habilitadas.
- Al volver a la actividad principal desde la actividad del mapa, se deberá mantener el listado en la posición en el que estaba antes de mostrarse la actividad del mapa, y las diferentes selecciones (de la cámara y del menú).

- En la actividad principal, mientras se está descargando la imagen de una cámara, el usuario no debería poder seleccionar otra cámara, por ejemplo, deshabilitando el listado de cámaras[1].
- Mientras se está descargando y procesando el fichero KLM, las funcionalidades de los iconos y del menú de opciones de la actividad principal deberán estar deshabilitadas, o bien estarán ocultos los iconos y el menú.
- Actualmente la codificación de caracteres del fichero *KLM* que se descarga de internet no es correcta, pues no se han usado las entidades apropiadas para las letras con tildes o las ñ. Por ese motivo observará que en la aplicación que desarrolle se muestran unos caracteres no imprimibles en algunos nombres de cámara. Deberá eliminarlos usando alguna expresión regular para que no aparezcan en el listado. Al eliminarlos algunos nombres de cámaras quedarán mal escritos, por ejemplo "Plaza de España" aparecerá como "Plaza de Espaa"

[1] Puede usar para ello el método setEnable() de ListView

Anexo

Este anexo le puede servir de ayuda para implementar algunas de las funcionalidades descritas en el enunciado. No están contemplados todos los aspectos que necesita para la realización de la aplicación, pues uno de los objetivos de esta práctica es que aprenda a buscar la información que necesite en la documentación que Google ofrece sobre Android.

Error con el protocolo http

Es posible que, en los laboratorios de la Escuela, al intentar acceder a un recurso usando el protocolo http, es decir sin certificado, se produzca un error de ejecución lanzando la excepción: `android java.io.ioexception cleartext http traffic to not permitted`.

Esto se soluciona añadidiendo en el manifest, en el elemento `<application>`, el atributo `android:usesCleartextTraffic="true"`.

Redirecciones http

Cuando se realiza la conexión a un servidor para solicitar un recurso mediante el protocolo HTTP, puede devolver el código HTTP_OK si el recurso existe, pero también alguno de estos otros códigos: HTTP_MOVED_PERM, HTTP_MOVED_TEMP o HTTP_SEE_OTHER. Con estos códigos se indica que el recurso ha cambiado de dirección, por lo que para conocer la nueva dirección hay que mirar en la cabecera de la petición de respuesta, donde se encontrará la nueva URL donde previsiblemente se encuentra el recurso. Es posible que incluso al realizar la conexión a esa nueva URL se devuelva de nuevo alguno de esos códigos y haya que consultar de nuevo la cabecera de la petición para obtener la nueva URL.

Para leer de la cabecera la nueva URL se dispone del método `getHeaderField("Location")` de la clase `URLConnection`.

Uso de la barra de herramientas y menús por los fragmentos

La información expuesta en este apartado es un resumen aclaratorio de lo documentación de Google: Fragmentos: Cómo trabajar con la barra de la app.

La barra de herramientas puede ser creada y usada solo por la actividad, creada y usada solo por un fragmento o creada por la actividad y usada tanto por la actividad como por los fragmentos de esta.

En la aplicación a desarrollar la barra de herramienta podría ser usada por los fragmentos, además de por la actividad, por ello lo recomendable sería que la creara la actividad y que los fragmentos que la quieran usar se registren para poder hacerlo. Por ejemplo:

- El ítem del *checkbox* de localización se puede gestionar en la actividad principal, de esta forma todo lo que sea pedir permisos lo haría la actividad principal.
- Los ítems *radiobutton* también los puede gestionar la actividad principal, que se encargaría simplemente en dejar marcado el *radiobutton* correspondiente
- Las opciones de recargar u ordenar las puede gestionar la actividad o el fragmento listado, dependiendo de donde se hayan implementado estas acciones. Por ejemplo, dado que la actividad es la que al inicio descarga el fichero KLM, lo apropiado sería que sea ella la que se encargara de esta acción..

El fragmento detalle necesita saber qué opciones de menú están marcadas antes de arrancar la actividad del mapa. Esta información la puede obtener accediendo a los ítems del menú para obtener cuales están seleccionado. Por tanto, también necesitaría acceder al menú, si bien no gestionaría los eventos que se producen al hacer clic en sus ítems.

De manera alternativa, la actividad y/o el fragmento listado podrían guardar en un objeto `ViewModel`, que compartan los fragmentos y la actividad, las selecciones de los ítems del menú. Si se hace así, el fragmento detalle no necesitaría acceder al menú, pues podría obtener la información de los ítems seleccionados consultando la información del objeto `ViewModel`.

Para que un fragmento pueda acceder a la barra de herramientas y al menú, creados por la actividad, necesita hacer lo siguiente:

- En método `onCreate()` invocar al método `setHasOptionsMenu(true);`
- Si quiere gestionar los eventos del menú, implementar el método *onOptionsItemSelected()*.
- Si quiere acceder a los ítems del menú, para por ejemplo ver el estado de marcado/desmarcado de algún ítem de tipo *checkbox* o *radiobutton*, necesita tener una referencia al objeto menú. Esto se puede conseguir implementando el método onCreateOptionsMenu*()*, ya que este método recibe como parámetro la referencia del menú y puede guardar esta referencia en un atributo. Pero observe que, en el caso de implementar este método en un fragmento, es un método diferente al de la actividad porque tiene 2 argumentos en vez de uno.

Los eventos del menú los pueden gestionar: únicamente la actividad, solo un fragmento, o bien tanto la actividad como los fragmentos. Esto depende del valor devuelto por el método *onOptionsItemSelected()* de la actividad:

- Cuando devuelve `true`, ya no se invoca al método *onOptionsItemSelected()* que podrían implementar los fragmentos.
- Si devuelve `false`, entonces el evento se propaga a los métodos *onOptionsItemSelected()* que implementen los fragmentos.

En moodle hay un ejemplo, llamado *Demo menús*, que tiene dos fragmentos y un menú compartido por la actividad y los fragmentos. En este ejemplo la actividad gestiona ella sola algunos eventos de los ítems, otros los gestiona uno de los fragmentos, y hay un evento de un ítem que lo gestiona tanto la actividad como un fragmento. Otro de los fragmentos no gestiona los eventos del menú, pero accede a ellos para ver qué ítems están marcados/desmarcados. Y también, tanto la actividad como los fragmentos comparten información de los clics que se han hecho en alguno de los ítems mediante un objeto `ViewModel`.

Hay que tener en cuenta que la ejecución de los métodos del ciclo de vida de la actividad y de los fragmentos, y del método `onCreateOptionsMenu()`, se hace en un orden determinado, por lo que un intento de acceder al menú sin que este haya sido creado, dará un error de ejecución.

Flecha de vuelta atrás

En la actividad del mapa debe mostrarse la flecha de vuelta atrás (a la actividad anterior) en una barra de menús. En la sesión 8 se vio como crear una `ToolBar` con la flecha de vuelta atrás y cómo gestionar el evento que se produce al hacer clic en este elemento dentro del método `onOptionsItemSelected()`. Pero ocurre que la actividad del mapa no tiene menús, con lo cual, para gestionar el evento hay que sobrescribir el método `onSupportNavigateUp()` para que invoque al método que se ejecutaría al pulsar el botón físico de vuelta atrás.

```
@Override
public boolean onSupportNavigateUp() {
   super.onBackPressed();
   return true;
}
```

Tal y como se vio en el ejercicio 3 (Barra de herramientas) de la sesión 8, para poder sobrescribir este método, y poder usar la `ToolBar`, la actividad debe heredar de `AppCompatActivity` en vez de `FragmentActivity`.

Fechas

- En Java se puede obtener la fecha y hora actual instanciando un objeto de la clase java.util.Date. Ejemplo: `Date fecha = new Date();`
- Una fecha se puede convertir en `String` mediante `String.valueOf(fecha)`, suponiendo que la variable fecha es una instancia de `Date`. El problema es que el valor almacenado en el `String` puede tener un formato que no es el deseado, por ejemplo: "Fri Apr 29 16:49:03 GMT+00:00 2022".
- Si se quiere obtener un `String` a partir de un objeto `Date` con un formato concreto, puede usarse la clase java.text.SimpleDateForm, indicando un patrón que determinará el formato de la fecha, y luego invocando al método `format()`. Por ejemplo, si se quiere obtener un `String` de la fecha actual en formato día/mes/año, sin información de la hora:

```
Date fecha = new Date();
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy");
String fechaString = sdf.format(fecha);
```

- Una fecha almacenada en un `String` puede convertirse en un tipo `Date` mediante el método `parse()` de un objeto `SimpleDateFormat`. Este objeto debe haberse creado con un patrón que coincida al formato con el que está almacenada la fecha en el `String`, si no, saltará una excepción. Por ejemplo, si la fecha es un `String` en formato año/mes/día, puede convertirse a un tipo `Date` mediante este código:

```
Date fecha;
String fechaString="2022/04/26";
SimpleDateFormat sdf = new SimpleDateFormat("yyyy/MM/dd");
try {
      fecha = sdf.parse(fechaString);
} catch (ParseException e) { e.printStackTrace();  }
```

Agrupación de marcadores en un mapa

Cuando en un mapa existen muchos marcadores, Google Maps ofrece la posibilidad de agruparlos en clúster, de forma que en cada clúster se indica el número de marcadores que hay dentro sin mostrar la ubicación de cada uno. Según se vaya haciendo zoom, los clústeres se van deshaciendo y mostrando, o bien otros clúster más pequeños, o bien los marcadores que contenía el clúster.

En esta página encontrará toda la información que se necesita para poder usar la agrupación de marcadores.

Accesibilidad

La accesibilidad en el sistema operativo Android es importante para garantizar que todas las personas, incluyendo aquellas con discapacidades visuales, auditivas, motoras y cognitivas, puedan utilizar las aplicaciones de manera efectiva. Para lograr esto, Android ofrece una amplia variedad de herramientas y opciones de accesibilidad que pueden ser activadas y personalizadas por el usuario.

Uno de los servicios de accesibilidad más importantes que se incluyen en Android es el *TalkBack*, un lector de pantalla que convierte el texto en voz y que puede ser activado en la configuración del dispositivo. *TalkBack* permite que las personas con discapacidad visual puedan navegar por las aplicaciones, leer mensajes y realizar otras tareas en sus dispositivos móviles.

Otro servicio de accesibilidad que incluye Android es Accesibilidad con interruptores. Mediante este servicio una persona con discapacidad motora puede navegar por la aplicación usando interruptores externos (que activa por ejemplo con el pie, la boca, etc.), usando los botones de bajar/subir volumen, o incluso la cámara realizando gestos (giñar un ojo, sonreír, etc.). Este servicio se configura asociando cada interruptor a las acciones *Siguiente*, *Anterior* o *Seleccionar*, de forma que la persona podrá navegar por la aplicación usando estos *interruptores* en vez tocar la pantalla. Se puede combinar con TalkBack de forma que además se reproduzcan el texto de cada elemento por el que se pasa.

Para que una aplicación en Android sea accesible, el programador debería tener en cuenta lo siguiente:

1. Utilizar **etiquetas de accesibilidad**: Es importante que se utilicen etiquetas de accesibilidad adecuadas para que los lectores de pantalla puedan leer y describir adecuadamente el contenido de la aplicación. De esta manera, las personas con discapacidad visual podrán utilizar la aplicación de manera efectiva. También ayudaría a cualquier persona en entornos *discapacitantes* como aquellos de oscuridad o mucha luz en los que apenas se ve la pantalla.
2. **Diseño adaptable**: El diseño de la aplicación debe ser adaptable y debe permitir que los usuarios puedan personalizar la aplicación para adaptarse a sus necesidades. Por ejemplo, los usuarios deben poder ajustar el tamaño del texto y cambiar los colores de la interfaz para mejorar la legibilidad.
3. **Usabilidad**: La aplicación debe ser fácil de utilizar y entender. Debe haber un flujo de navegación claro y sencillo y el contenido debe estar bien organizado para que los usuarios puedan encontrar lo que están buscando.
4. **Texto alternativo**: Es importante proporcionar en la aplicación texto alternativo para las imágenes y otros elementos visuales. Esto permitirá que los lectores de pantalla describan adecuadamente los elementos a los usuarios con discapacidad visual.
5. Soporte de **subtítulos** y **audio descripción**: Si la aplicación contiene video, es importante proporcionar opciones de subtítulos y audio descripción para que los usuarios con discapacidad auditiva o visual puedan entender el contenido.
6. **Pruebas de accesibilidad**: Es importante realizar pruebas de accesibilidad en la aplicación para asegurarse de que cumple con los estándares de accesibilidad. Esto se puede hacer utilizando herramientas de pruebas automatizadas y realizando pruebas manuales con usuarios con discapacidad.

En resumen, el programador debería tener en cuenta estos aspectos para que su aplicación en Android sea accesible. Al tener en cuenta estas consideraciones, se puede garantizar que la aplicación sea utilizada por una audiencia más amplia, incluyendo personas con discapacidades. Además, esto puede mejorar la calidad general de la aplicación y hacerla más fácil de usar para todos los usuarios.

En https://developer.android.com/guide/topics/ui/accessibility hay más información acerca de cómo desarrollar aplicaciones accesibles o como probar si una aplicación es accesible.

Algunas sugerencias para hacer accesible la aplicación de Cámaras de Madrid:

- Usar un contraste de color adecuado. A continuación, verificarlo instando la aplicación Prueba de accesibilidad.
- Describir cada elemento de la UI, usando el atributo `contentDescription`, para que un lector de pantalla pueda anunciar la existencia de cada elemento. Se deben usar descripciones adecuadas

y fáciles de entender. Como las fotos de las cámaras no se pueden describir de forma particularizada, se pueden usar descripciones del tipo "Foto de la cámara de la calle XXXX".

- Usa la aplicación *TalkBack* para experimentar cómo un discapacitado visual, que usa un lector de pantalla, sería capaz de interpretar la información que escucha para poder interactuar con la aplicación.
- Al arrancar podría mostrar un cuadro de diálogo que pregunte al usuario si quiere activar la accesibilidad de la aplicación. Si responde afirmativamente, la interfaz podría ser algo diferente para favorecer a las personas con algún tipo de discapacidad, por ejemplo:
    - Mostrar los textos más grandes.
    - En el listado, mostrar solo 5 cámaras[2].
    - Debajo del listado de las cámaras se pueden poner unos botones en forma de flecha izquierda y flecha derecha de forma que cuando se pulsen se haga *scroll* de 5 elementos. Y es que para ciertas discapacidades es más accesible usar botones que un *scroll*[3].
    - Debajo de la foto, en la parte izquierda, poner el símbolo de ampliar para que al pulsarlo la foto ocupe toda la pantalla, de forma que las personas con baja visión puedan visualizarlas mejor.
    - Añadir un botón para permitir que el usuario de órdenes con la voz, por ejemplo, manejarse por la actividad del listado mediante comandos "siguiente", "anterior", "ordenador alfabéticamente", "ver cámara", "abrir mapa", etc. o en la actividad del mapa: "ampliar", "reducir", "mover a la izquierda/derecha/abajo/arriba", … El reconocimiento de voz se puede hacer con el motor TextToSpeech de Android.
    - En la actividad del mapa:
        - Agregar un texto para describirlo. Consulta la página de Mejora la accesibilidad en los mapas de Google.
        - Poner unos botones grandes para hacer zoom.

Tras implementar esta interfaz accesible, activar la Accesibilidad con interruptores y configurar interruptores con la cámara para entender como un discapacitado físico que no tenga habilidad para usar la interfaz táctil de la pantalla se podría mover por la aplicación.

[2] Usar un número reducido de cámaras en el listado permite que los textos puedan ser más grandes, con lo que se ayuda a personas con dificultades de visión u otra discapacidad física. También a las personas que usen un lector de pantalla, al no tener que escuchar un listado muy largo y poder volver a los nombres de las cámaras del listado si lo necesita. La elección del nº 5 es por la analogía a los dedos de la mano

[3] A una persona que necesita un puntero para interactuar con la pantalla le facilita disponer de dos botones para desplazarse entre los bloques y luego señalar la cámara que quiere.