Proyecto Final - Desarrollo de Aplicaciones Móviles en Plataforma Android

Sergio Morales Universidad Cenfotec

Descripción

El proyecto consiste en una aplicación simple para el reporte de averías en carretera. Debe proveer una interfaz fácil de usar que permita al usuario observar todas las averías reportadas por otros usuarios (recuperadas desde un endpoint web), y editarlas, eliminarlas o crear nuevas (haciendo uso de metodos en el mismo servicio web). Las averías deben ser visibles tanto a modo de lista como a modo de puntos en un mapa, desplegado usando Google Maps. Al reportar una avería, se puede tomar un foto, que se va a asociar a la avería en el backend de la aplicación.
Este proyecto representa un 60% de la evaluación final del curso.

Pantallas y Flujo de Uso

- La **pantalla de login** ofrece un campo de usuario y contraseña, y un botón de registro. El botón de registro debe llevar a un **formulario de registro** para que el usuario se registre especificando su nombre, correo, teléfono y número de cédula, así como un usuario y un password. Estos deben ser ingresados a una base de datos SQLite local en el dispositivo Android y usarse para validar entradas a la aplicación desde la pantalla de login.
- La **pantalla de averías** muestra todas las averias, con 2 "tabs" en la parte superior de la misma, que sirven para cambiar entre la lista de averias, y el mapa con todas las averías representadas como íconos dentro del mismo. Al tocar una avería dentro de la lista, se pasa a la **pantalla de detalles**. Al tocar un ícono dentro del mapa, se muestra un **diálogo de información** asociado a ese ícono, mostrando datos básicos del mismo. Desde este diálogo se puede proceder a la **pantalla de detalles**. En modo de lista, debe existir un **botón de creación de avería**. En modo de mapa, si un usuario realiza una acción de "longpress" sobre un lugar no ocupado del mapa, esto debe generar un **diálogo** que pregunte si el usuario desea crear una nueva avería en ese punto.
- La **pantalla de detalles** muestra una descripción detallada de cada avería, con todos los datos disponibles a través del endpoint, incluyendo la imagen, si existe. Desde esta pantalla, puede procederse a la pantalla de edición.

- La **pantalla de edición** es un simple formulario con el que se pueden modificar todos los valores de la avería consultada. Esto incluye la foto de la misma. Esta pantalla también provee una opción de eliminación de avería, detrás de la cual debe haber un diálogo de confirmación.
- La **pantalla de creación** de avería es muy similar a la de edición y consiste en un formulario y un espació para agregar una foto a la misma

Herramientas y Prácticas a Utilizar

- **Butterknife** para la inicialización de elementos de interfaz
- **Retrofit** para el contacto con el servicio web
- **Picasso** para cargar imagenes en elementos de interfaz para los diálogos de confirmación
- El patrón de **Adapter** para la implementación del **RecyclerView** para la lista de averías
- El **API de Imgur** (https://apidocs.imgur.com/) para subir imagenes a dicho servicio
- **ViewPager** para la interfaz de tabs en la pantalla principal
- **Google Maps** para la funcionalidad de vista de mapas
- La **aplicación de cámara** para la toma de fotografías
- **Arquitectura MVC** para la organización de clases y paquetes
- **ORMLite** para la gestión de la base de datos
- **SharedPreferences** para recordar los datos de usuario ingresados en las pantallas de edición y creación de usuario

Documentación del API

1. **NO TENGAN MIEDO DE EXPERIMENTAR CON ESTE API**. No hay manera de arruinarlo, entonces siéntanse en todo el derecho de crear nuevas averías, borrarlas, editarlas, etc con tal de entender tanto como puedan acerca de como este API funciona. Lo peor que puede pasar es que borren todas las averías, en cuyo caso solo crean un par de averías nuevas y ya, o me mandan un correo y yo las creo. En serio, no se detengan.
2. La raiz del API es: **https://fn3arhnwsg.execute-api.us-west-2.amazonaws.com/produccion**
3. Las llamadas disponibles en este API, y que van a tener que implementar como parte del proyecto, son las siguientes:
    - **GET /averias**: Retorna una lista de averias. Cada una de estas averias tiene un ID, nombre, descripción, tipo y ubicación

- **GET /averias/<ID> donde <ID> es el ID de una avería**: Retorna los detalles de una avería. Esta es la unica manera de obtener los datos del usuario de la avería, así como la fecha de reporte, por lo que no basta con el metodo anterior para ver los detalles de una avería, también tienen que usar este.
- **POST /averia**: Este método ocupa que el body sea una estructura de JSON de una avería (ver abajo), y lo que hace es crear una avería nueva. El ID de esta avería debe ser único. Si ya hay una avería con el ID provisto, la llamada falla.
- **POST /averia/<ID> donde <ID> es el ID de una avería**: Este método ocupa que el body sea una estructura de JSON de una avería (omitiendo el parámetro de id) y modifica los datos de una avería que ya existe. Obviamente, el <ID> provisto debe de pertenecer a una avería que ya exista, o la llamada falla.
- **DELETE /averia/<ID> donde <ID> es el ID de una avería**: Borra a la avería con el <ID> provisto de la base de datos.

Esos son todos los metodos que el API expone. Cualquier otro metodo va a fallar.

4. Para referencia, este es un ejemplo de un objeto de Averia:

Listing 1: Ejemplo de objeto JSON para una avería.

```
{
 "id":"123",
 "nombre": "Averia Ejemplo",
 "tipo": "SEMAFORO",
 "usuario": {
    "correo": "sergiome@gmail.com",
    "nombre": "Sergio Morales",
    "tel": "88854764",
    "cedula": "113870567"
 },
 "fecha": "05/09/2017",
 "descripcion": "Este es un ejemplo",
 "imagen": "URL IMAGEN",
 "ubicacion": {
    "lat": 12,
    "lon": 20
 }
}
```

Recuerden que como es un objeto JSON, el orden de los elementos no importa. Solo que la estructura sea la misma y los nombres de los campos también (los valores no, obviamente).

5. Este API esta protegido por un API KEY. Para poder usar cuaquier llamada, en el header deben incluir un campo **con la llave "x-api-key" y el valor de llave especificado abajo**. Recuerdar que en retrofit, esto significa que todas las llamadas deben estar anotadas con un campo de header que se vea así:

Listing 2: Ejemplo de declaracion de Headers en RetroFit.

```
1 @Headers("x-api-key: rabArf10E86thWRQ5u4MH3pFXVpiQiXv8jg1c4hO")
2 @GET("averias")
3 Call<List<Averia>> obtenerListaDeAverias();
```

Recomiendo meter el api key en una variable para no estar copiando y pegando el mismo String para las 5 llamadas.

Extras

Se considerará a criterio del profesor cualquier adición que se le haga al proyecto sobre lo especificado en este documento, por ejemplo:

- El uso del API de animación de Android.
- Manejo de errores consistentes a través de la aplicación.
- Diseño de UI/UX atractivo (uso efectivo de NavigationBar, ViewPagers, etc).

Entrega y Evaluación

El proyecto debe ser entregado y demonstrado el día de la última clase del curso en sesiones 1 a 1 con el profesor. Este debe ser realizado y evaluado de manera individual.