Práctica 2 – Android: contacto con API remoto

Introducción

El objetivo de esta práctica es aprender a comunicarse con un API remoto desde una app Android, y mostrar en la app los resultados obtenidos del API de forma eficiente empleando `RecyclerView`. Para esta práctica es **obligatorio conectarse al API usando `AsyncTaskLoader`** como se describe a continuación, y no se permite utilizar librerías como Volley u OkHttp para conectarse al servicio.

La app desarrollada mostrará una interfaz sencillo para realizar búsquedas sobre el API de Google Books. La interfaz permite buscar libros en base al nombre de los autores y el título del libro, y buscar únicamente libros, revistas o ambos. Los resultados se muestran en una lista utilizando `RecyclerView`, mostrando para cada resultado el título, el nombre de los autores, y una URL con información más detallada.

Instrucciones

Crea un nuevo proyecto Android usando la plantilla “Empty Activity”. Corre la app en el emulador para comprobar que la plantilla se ha creado correctamente.

- Añade `implementation 'com.google.android.material:material:1.0.0'` a build.gradle (module app) en la sección `dependencies`. Acepta la sugerencia “Sync now” de Android Studio.
- Modifica la layout file de MainActivity para crear una interfaz similar a las imágenes de abajo. Sugerencia: utiliza `ConstraintLayout` y `RadioGroup`.
- En un navegador examina el API de Google Books, en particular la operación volume.list que permite buscar libros en Google Books usando una consulta de texto.
- Crea una clase `BookLoader extends AsyncTaskLoader<String>` que devuelve en su método `loadInBackground` el resultado de invocar a la operación volume.list del API de GoogleBooks, para unos `String queryString, String printType` dados.
  - En el método `onStartLoading` de `BookLoader` llama a forceLoad() para forzar siempre la carga del resultado, y no utilizar resultados de búsquedas anteriores.
  - Implementa un método `String getBookInfoJson(String queryString, String printType)` en la clase que consideres oportuna, que contenga el código para hacer la petición al servicio usando `HttpURLConnection`.
  - Para utilizar `BookLoader` desde `MainActivity`
    - Crea una clase `BookLoaderCallbacks implements LoaderManager.LoaderCallbacks<String>` y añade una variable `private BookLoaderCallbacks bookLoaderCallbacks = new BookLoaderCallbacks()` a `MainActivity`. El método `onCreateLoader` de `BookLoaderCallbacks` devolverá una nueva instancia de `BookLoader`.
    - En `onCreate`, inicializa el loader. Usando las clases de Jetpack `androidx.loader.app.LoaderManager`, `androidx.loader.content.AsyncTaskLoader`, `androidx.loader.content.Loader`, se puede hacer con el código siguiente:

```
LoaderManager loaderManager = LoaderManager.getInstance(this);
if(loaderManager.getLoader(BOOK_LOADER_ID) != null){
    loaderManager.initLoader(BOOK_LOADER_ID,null, bookLoaderCallbacks);
}
```

- Añade un método `public void searchBooks(View view)` a `MainActivity` y regístralo como event handler del botón con la propiedad `android:onClick`. En `searchBooks` lanza el loader con un código similar al de más abajo.

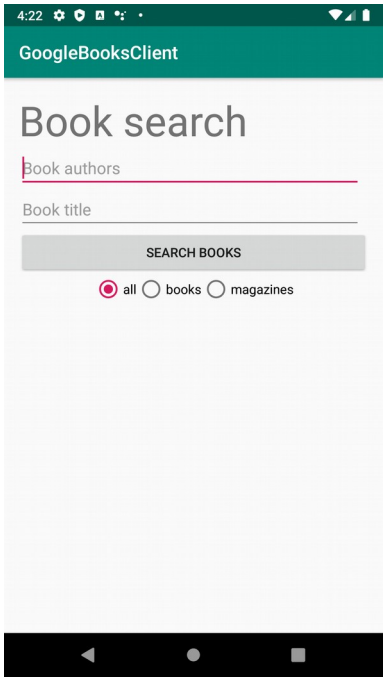
```
Bundle queryBundle = new Bundle();
queryBundle.putString(BookLoaderCallbacks.EXTRA_QUERY, queryString);
queryBundle.putString(BookLoaderCallbacks.EXTRA_PRINT_TYPE, printType);
LoaderManager.getInstance(this)
       .restartLoader(BOOK_LOADER_ID, queryBundle, bookLoaderCallbacks);
```

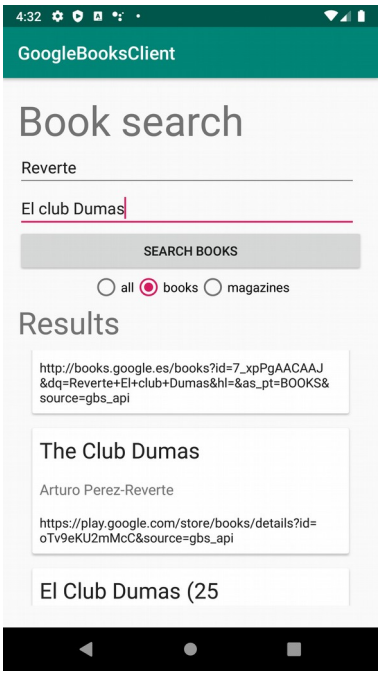
  - En ese código `queryString` es la concatenación de los nombres de los autores y del título del libro tal y como se introdujeron en las `EditText` del interfaz de usuario, añadiendo un espacio entre medias.
  - Para `printType`, utiliza el método `getCheckedRadioButtonId()` de `RadioGroup` para determinar que `RadioButton` está activo.
- Sugerencias:
  - Lee Making an HTTP connection para un ejemplo de como realizar una petición a un servicio. No olvides añadir los permisos necesarios para que la activity tenga acceso a Internet.
  - Consulta este ejemplo de como utilizar `AsyncTaskLoader` para realizar una petición a un servicio con `HttpURLConnection`.
  - Muestra el string devuelto por el servicio en Logcat para comprobar que la app contacta con el servicio correctamente. Utiliza el parámetro `maxResults` de la petición para limitar el número de resultados devueltos, y que sea más fácil de entender el JSON devuelto

- Define una clase `BookInfo` para guardar la información sobre los resultados de la búsqueda, con campos `String title; String authors; URL infoLink`. Convierte el string en formato JSON devuelto por el API a una lista de objetos `BookInfo` en un método `static List<BookInfo> fromJsonResponse(String s)` de `BookInfo`. Modifica la clase `BookLoader` para que ahora extienda `AsyncTaskLoader<List<BookInfo>>`.
  - Sugerencias:
    - Lee Parsing the results y este ejemplo para ver cómo manejar JSON.
    - Utiliza Logcat para comprobar que el string JSON se está parseando correctamente.
- Crea una clase `BooksResultListAdapter extends RecyclerView.Adapter<BooksResultListAdapter.ViewHolder>` que sirva de adapter de la recycler view, siguiendo lo visto en clase de teoría.
  - Usar `androidx.cardview.widget.CardView` como en la imagen de ejemplo es opcional, se puede también utilizar un interfaz más sencillo
  - Sugerencias:
    - Lee RecyclerView para ver ejemplos de uso de RecyclerView
    - Añade una variable `private ArrayList<BookInfo> mBooksData` a `BooksResultListAdapter` que contenga los datos a mostrar. Añade un método `public void setBooksData(List<BookInfo> data)` para modificar esta variable.
    - Añade un método `void updateBooksResultList(List<BookInfo> bookInfos)` a `MainActivity` para actualizar los datos de la recycler view, que llame a los métodos `setBooksData` y `notifyDataSetChanged()` de `BooksResultListAdapter`.

        - Para comprobar que todo funciona bien, llama a `updateBooksResultList` con una lista cualquiera en `onCreate`, para ver que se visualiza la lista correctamente en el interfaz.
- En BookLoaderCallbacks llama a updateBooksResultList con el lista que se ha cargado cargada.
    - En `getBookInfoJson` pasa el valor 40 en el parámetro maxResults del [API de GoogleBooks](), para obtener el máximo de resultados posibles del API.

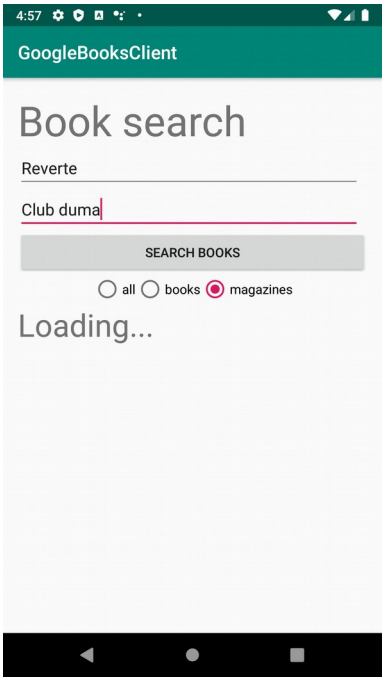
El resultado final deberá tener un aspecto similar a las siguientes pantallas:

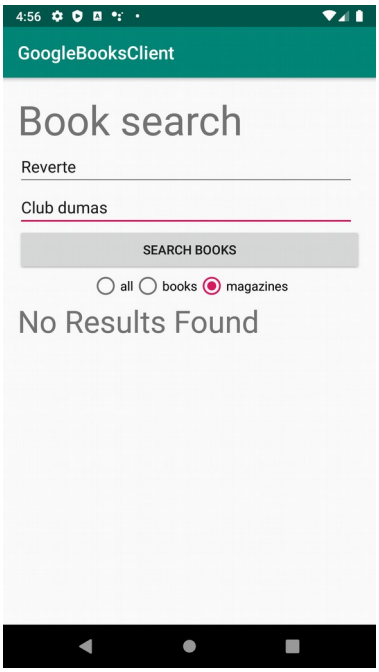




Opcionalmente

- Usar `androidx.cardview.widget.CardView` como en la imagen de ejemplo, para las entradas de la `RecyclerView`.
- Añade un event handler para onclick de las entradas de la `RecyclerView`, que abra la URL de detalle de la entrada para abrirse en el navegador, utilizando un `Intent` implícito.
- Mientras se realiza la búsqueda, muestra el texto “Loading...” en vez de “Results”, para dar una indicación de que la aplicación está trabajando para resolver la búsqueda, como se ve en la imagen de más abajo.
- Si una búsqueda devuelve cero resultados, muestra el texo “No Results Found” en vez de “Results”, como se ve en la imagen de más abajo.





A enseñar durante la corrección

1. Lanzar la app en el emulador:
   - El interfaz de usuario es similar al de la imagen del enunciado.
   - Se pueden buscar libros en el API de Google Books y está implementado usando `AsyncTaskLoader` y mostrando los resultados en una `RecyclerView`
2. Mostrar las partes opcionales que se han implementado