Hi,

The following are my suggestions to further improve the app:

- Gson should be used to parse JSON data. It has far superior performance.
- RecyclerView should be used to display lists of data
- From a UI/UX perspective, I would suggest a more Material Design theme - as this is what android users often expect - however, if that theme is required for branding purposes then it is fine.
- ButterKnife could be used, however I do not think it is necessary as API 26 means that casting is no longer required in many cases with findViewById, so it is not as syntactically clunky as it was in the past.
- To support many international versions, you could use product flavours and build variants to support your requirements. Within the build.gradle I have set the resConfig to auto, to allow for multiple language fallbacks.
- For data storage, I would suggest moving away from an SQLite implementation. A good solution could be realm. In addition to providing a database, Realm provides a data-driven backend solution that can drive your business goals.
- The Songs are Serializable; they could be moved to Parcelable for a performance gain.
- RxJava can replace the AsyncTasks within the app. It would be syntactically cleaner, and require less maintenance; in addition, RxJava provides a number of capabilities that allow you to make sophisticated network calls - depending on your backend and business requirements. RxJava excels in filtering data sets, and this app uses filters; making Rx an ideal solution.
- Architecturally I began a Model View Presenter implementation. However, further refactoring is required in order to be fully compliant with MVP.
- I began implementing Dagger 2, to highlight that the Database could be provided using dependency injection.
- The app uses Activity, not AppCompatActivity. AppCompatActivity is the way to go because the ActionBar is not encouraged anymore and you should use Toolbar instead (that's currently the ActionBar replacement).
- Should the app’s feature set be extended to include a detail screen or other screens, then I would suggest moving towards Fragments; as they allow for a faster and more dynamic navigation.
- Kotlin is now an official language by Google, and is a solid option to develop apps in; going with Kotlin would drastically reduce the amount of boilerplate code.

Regards,
Bertrand