Android App Improvement

MVP, Continuous development

https://github.com/patrick-doyle/android-rxmvp-tutorial

Core Improvements

* UI
* Unit Testing
* Reusability
* Standardization

UI

**BlueTooth**

There are libraries out that that can simplify the Bluetooth all the boiler plate code, allow for development time to be spent on passing and handling the data. There is a library that uses the RxJava api for it’s Bluetooth API which would allow for reactive programming to recover and handle exceptions gracefully. Also this library has unit testing in mind so this can become testable through the continuous build process.

Improvements

Issues

**Association Data references in code.**

There seems to be some code that references the Associations.



**Unused variables and Commented Code**

Variables and commented code that is not used can be deleted as it will is saved within the repository and causes confusion reading through the code base.

**Updating UI from Business Logic layer**

AsyncTasks are executing UI changes within their class which should be considered the business layer. Calling UI changes from the AsyncTask method causes confusion as the UI changes are happening where the business logic should only be happening. Once an AsyncTask completes it’s network call and business logic it should use a listener to pass an object representing the retrieved and transformed data back to the UI component for further View manipulation.

**Use of JSONObject**

Data retrieve from the server is returning a JSONObject from the response. This adds a lot more boiler plate code as you must check if the specific variable you are looking for is within that object causing many nested if/else checks cluttering the code. With open source libraries we can simplify the process of AsyncTasks while also maintaining Type Safety. GSON is a 3rd part library created by Google which consumes JSON from response and transforms them into a declared Java object. This can be tied together with Refrofit which is another open source library created by Square which allows you to neatly create interfaces to call API endpoints which are then returned as the desired typed safe object of the API request.

**Recursive Loops**

There are some recursive loops within the code. The Initialize Activity gets called recursively during the beginning of the app life cycle if certain conditions are not met. This prevents the app from closing gracefully which can be a bad experience for the user and possible drain their battery.

**Improper use of Multilingual**

Android is designed to support multiple languages within the strings.xml resource files within it’s resource directory. The android application will determine which language is need based off of the user’s locale. Currently translations are being persisted to the database during initialization after they are retrieved from the server. Every screen that calls for translations makes a database call from the UI tread, increasing the change of an Application not Responding error. The Android framework does more then just handle multilingual text but also supports RTL and LTR text formats for cultures that use them.

**Excessive Database Calls**

It appears there is a lot of database calls from the UI thread specially for the language entries. Database calls are an expensive resource and should always be optimized. I believe database calls should be also called from a thread outside the UI thread as if the database takes awhile to return you are going to hang up the UI thread and possible have an Application Not Responding.

**Libraries**

NetWorking

* OkHttp
  + <https://github.com/square/okhttp>
* Retrofit
  + <https://github.com/square/retrofit>
* Glide
  + https://github.com/bumptech/glide

Dependency Injection

* ButterKnife
  + <https://github.com/JakeWharton/butterknife>
* Dagger
  + <https://github.com/google/dagger>

JSON Parsing

* GSON
  + <http://google-gson.googlecode.com/svn/trunk/>

Database Management

* ObjectBox
  + http://objectbox.io/

BluTooth

* RxAndroidBle
  + https://github.com/Polidea/RxAndroidBle

Logging

* Timber
  + <https://github.com/JakeWharton/timber>

Crash Reporting

* Firebase
  + https://firebase.google.com/docs/crash/android

Reactive Programming

* RxJava
  + <https://github.com/ReactiveX/RxJava>
* RxAndroid
  + <https://github.com/ReactiveX/RxAndroid>

Leak Detection

* LeakCanary
  + https://github.com/square/leakcanary

Utility

* <https://github.com/google/guava> (general utility)

Unit Testing

* Junit
  + <http://junit.org/junit5/>
* Mockito
  + <http://site.mockito.org/>
* Expresso
  + <https://developer.android.com/training/testing/espresso/index.html>