**Backendless Help Guides**

**Part 1: Setup Backendless, follow their Getting Started Guide**

Find the Android getting started guide here (the steps below refer to this guide):

<https://backendless.com/mobile-developers/quick-start-guide-for-android/>

Example app for download at [www.nemoquiz.com](http://www.nemoquiz.com) (CSA page): “Backendless\_save\_retrieve.zip”

* Create an account at backendless.com
* Create an app on the website and name it “CollegeApp”
* Follow the getting started guide. For step 2, don’t create a dependency in project structure. Instead, down the file (described in next step)

<https://backendless.com/mobile-developers/quick-start-guide-for-android/>

* + Download the “backendless.jar” file. Download here: <https://backendless.com/downloads/> (Get the android file, open the zip, find backendless.jar)
	+ Copy backendless.jar in to your project’s app/libs folder. Close and restart Android Studio.
* Follow the guide up to step 8.
* Steps 9-10 are optional. They create a new user that will show up in the Users table in your online account. CollegeApp will not use the Users table for anything, but if you want you can do these steps to test whether your app is able to connect to Backendless. If so, delete those lines of code after the test.
	+ We would dig into Users if the app had a login screen that checked passwords.
* Steps 11-13 create a “Comment” class, then create a Comment object to insert into the backendless database. This is optional as a test. The CollegeApp will store Profile objects instead of Comment objects.

**Asynchronous Tasks Concept**

Backendless database methods must be run in the background (i.e. asynchronously), or your app will crash. They cannot run in the User Interface (UI) Layer. Android doesn’t like it when slow processes run in the UI layer, so it forces you to run them in the background / in their own thread.

The example code for saving a Comment object uses an asynchronous callback function, which means it runs in the background and then executes some code when it finishes. The callback is the code that executes at the end. The Persistence.save() function takes two arguments, and the second of those arguments is the callback (the first is the object to be saved).

Backendless has its own callback class: BackendlessCallback. If you start typing “new BackendlessCallback,” you can hit enter to generate the structure of that code block. That will include and override for the “handleResponse()” method. The code inside that method is the callback code that will execute when the data is done being saved to the cloud. Pay attention to the argument listed inside the handleResponse() method – that is the data that backendless sends back to the app, and it can be used within the method.

You can observe how long it takes to save the data if you watch the time delay when the status TextView changes to “Saving data to Backendless…” and then changes again to the successfully saved message. If you are fast enough to mess with any views on your device’s screen in this short time, you will find that the UI layer is still responsive while the Persistence.save() call is working.

**Part 2 – Retrieve Data from Backendless**

You can use “Persistence.of(Comment.class).find()” to retrieve a list of Comment objects from the database. However, that will crash your app unless you run it asynchronously (in the background).

Inside the parentheses for “find(),” start typing “new BackendlessCallback” and hit enter when it recognizes the name. It will automatically generate the structure to override the handleResponse() method.

The argument to handleResponse() is a BackendlessCollection object called “response” that contains Comment objects. The Comment objects can be accessed using response.getData(), which returns a List<Comment> object. This List has a get() method that gives individual Comment objects. You could use “response.getData().get(0)” to get the most recent comment. The example app loops through the List object and displays each comment’s message in the status TextView.

When Backendless runs the find() command, it only returns a collection of the 10 most recent records by default. This is so the server doesn’t end up returning millions of records from a large database table. There are methods like BackendlessCollection.nextPage() for getting additional records.

Find the API documention for Android to see more of the search functions in backendless.

<https://backendless.com/products/documentation/>

Look in the “Basic Search” section under “Data Service.”