ACTIVITY **3.1.4**

Listing Trips

INTRODUCTION

In the previous activity, you created a new trip class and saved trips in the BaaS, Backendless. In this activity, you will create a query and search your Trip table in Backendless to create a list of all trips in your app.

Materials

- Computer with Android™ Studio
- Android™ tablet and USB cable, or a device emulator
- Free Backendless account per student

RESOURCES

Lesson 3.1 Reference Card for Backendless
Resources available online

Procedure

Part I: Comment TripListFragment Methods

1. Open your TripTracker app in Android Studio.

NOTE

If you were unable to complete *Activity 3.1.3 A New Trip,* open ***`3.1.3TripTracker_Solution`*** as directed by your teacher. Recall that if you import the solution, you must update keys values in ***`strings.xml`*** Specifically,

- Change `be_app_id` your Backendless App ID key value
- Change `be_android_api_key` to your Android API key value

You can retrieve these from your **Backendless Console** (**Manage** icon).

2. Document the following methods in `TripListFragment`. Because you are documenting methods with existing functionality, feel free to run the app to determine what some of the methods do.

Method	Comments
`onCreate`	
`onCreateView`	
`onListItemClick`	
`onCreateContextMenu`	
`onContextItemSelected`	
`onCreateOptionsMenu`	
`onOptionsItemSelected`	

3. Get a copy of *3.1.4TripTracker_StarterCode* from your teacher and copy or extract the files to a location that is convenient, but *not* in your project folder. Your Desktop is an example of a good location.
4. Copy and paste ***fragment_trip_list.xml*** into your ***res/layout*** folder in Android Studio. The functionality of the TripList screen that uses this layout is currently commented out, but the layout is used to display a single trip in a list of trips.
5. Open ***TripListFragment.java***. Search for **mTrips** and review all occurrences.

   What does `mTrips` represent?

Part II: Trip Ownership

6. Log in to the Backendless Console and open the **User** table. Compare objectId and ownerId values.
7. Open the **Trip** table. Compare objectId and ownerId values.

   Who owns each entry in the **User** table? How can you tell who owns each entry in the **Trip** table?

The objectId and ownerId are automatically created and given values by Backendless. To uniquely identify data items in the database, Backendless makes sure that each item has a unique objectId. For trip ownership (ownerid), Backendless assigns the objectId of the *user* to your trip. In TripTracker, you could use the user's email field to designate ownership like you did in CollegeApp, but you will use ownerId. It is a built-in ownership feature and may be useful in future apps, where you may not want to use a user's email address.

Part III: Complete the TripList/My Trips Screen

As you did in Activity 2.2.3, you will use an `ArrayAdapter` to display each trip's data in the List view. Your `TripAdapter` extends `ArrayAdapter` and overrides `getView`.

8. In `TripAdapter` `getView` (in `TripListFragment`), find the TODO comment. Remove the `return null` statement and define the contents of each row in the List view. Refer to *Activity 2.2.3 ListView* and its presentation for reference.

   `getView` should define `convertView` (one of the parameters) by inflating the XML file ***fragment_trip_list_item.xml***. Recall that the XML file defines the view for each row of the List view. In this case, it contains the name of the trip, the start date, and an edit icon.

   a. Get the `Trip` from the list using `getItem(position)`.
   b. Create two `TextViews`, one for the name of the trip, one for the start date.
      i. To create the `TextView` for the trip's name, use the `findViewById` method on `convertView` to define `trip_list_item_textName`.

ii. To create the `TextView` for the trip's start date, use the `findViewById` method on `convertView` to define `trip_list_item_textStartDate`.

iii. Use the Trip's `getName` and `getStartDate` to set the values for the `TextViews`.

iv. Because the edit icon is unrelated to your Trip data, it does not need to be referenced anywhere other than the XML file.

With `getView` defined in your custom Trip adapter, you now have a defined layout for each item in the array of trips to be shown in the `ListVew`. You still need to get trip data from Backendless, put it in an array, and notify the adapter that it has new data to display. This is all done in the `refreshTripList()` method.

9. Find the `refreshTripList()` method near the end of the file.

   a. Use the Backendless `find` method to retrieve all of the user's trips.

      - Recall that a user's trip is identified by the `ownerID`.

   b. Create a "where clause" to find trips owned by the current user, specifically, where `ownerId` equals `user.getObjectId()`. If necessary, refer to your CollegeApp for how to perform this type of find query.

   c. Override the `handleResponse` method and populate the `mTrips` ArrayList, iterating over the list of trips (`List<Trip>`) that is returned from the database. Again, refer to your CollegeApp if necessary.

   d. After you have populated `mTrips`, notify the adapter there is new data to display:

```
1: ((TripAdapter)getListAdapter()).notifyDataSetChanged();
```

   e. Override the `handleFault` method to log any error message that Backendless may encounter.

10. Test your app. Because you created some trips in the last activity, you should see those trips listed in the TripList/My Trips screen.

11. (If you have already created multiple users with their own trips, skip to the next step.) Create another user and some trips while logged in as that user.

12. Test your app further and confirm that the only trips in the list are those created by the user who is currently logged in.

Part IV: Update TripList

After a user saves a trip, the app should return to the TripList/My Trips screen to see their new trip in the list.

13. In `TripFragment`, when the `save` has finished, indicate that this Android activity is complete and direct the user back to the previous Android activity (the TripList activity), with:

```
1: getActivity().finish();
```

You may have noticed that every time you create a new trip, you need to touch **REFRESH** in the action bar to see the new trip. You can make this happen automatically.

14. When `TripListFragment` resumes, refresh the trip list:

```
1: @Override
2: public void onResume() {
3:     refreshTripList();
4:     super.onResume();
5: }
```

With the additions in this Android activity, your app has the ability to create and retrieve trips. Activity 3.1.5 will show you how to update and delete existing trips.

CONCLUSION

1. Review the comments and code from your `onCreate` Method.

```
// Create the Adapter that will control
// the ListView for the fragment.
// The adapter is responsible for feeding
// the data to the list view.

TripAdapter adapter = new TripAdapter(mTrips);
setListAdapter(adapter);
```

   Restate what the code does and how the adapter knows how to display its data.

2. Why did you use a user's `objectId` in place of email to identify a user in TripTracker?