EAD2 CA2 – Android Project

Tud tallaght campus | Main St, Tallaght

An album tracker app (albumtrackr)

ryan deering x00144631 & james lynam x00073019

2021

Table of Contents (Page Numbers will change)

**IntroductionX**

**GitHub InformationX**

**Database SchemaX**

**ServiceX**

**Application BreakdownX**

**MiscellaneousX**

**ConclusionX**

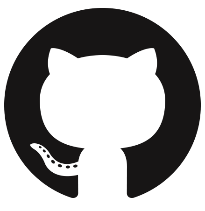
**Introduction**

For our second CA in Enterprise Applications Development 2, we decided to create an app that stores lists of an individual’s favourite albums into something like a playlist. The user will be able to create a list by assigning it a title and description, and once the list has been created, they would be able to add their favourite albums to this list.

When being prompted for input on the album, the [Last.fm API](https://www.last.fm/api) would be queried, and if the album was real and present in their system, it would be added to the database. The user will also have the ability to scroll through different tabs based on different circumstances. These tabs would allow them to see their favourite album lists (through the use of a star method), the most popular album lists and the newest album lists.

There will also be some validation that would ensure that users can only delete and change their own lists, and we hope to accomplish this through the use of the device ID. The Web API methods will be created in ASP.NET Core and the client side of the application will be using Java in Android Studio.

We both have had a long history of interest in music, and believe that this interest is making this CA more engaging.

**GitHub Information**

Repo URI: <https://github.com/ryandeering/albumtrackr.git>  
  
Need to add this once we’ve finished commits.

**Database Schema**

This page will have images and descriptions if the database and JSON schemas.

This section will contain short descriptions of operations, a screen shot of the Swagger UI test pages and how we have deployed the app – Azure etc.

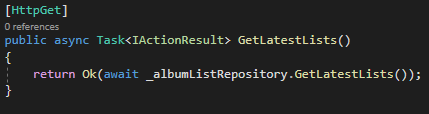
**Service**

**Our Operations**

**GET Methods:**

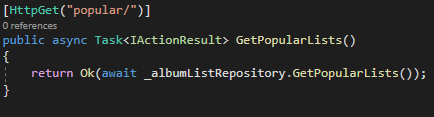
**GetLatestLists():** This GET method returns the **latest** album lists that were added to the service. It does this by ordering the list in a descending method, using the **Created** attribute present in the **AlbumList** class.

**/api/AlbumList**



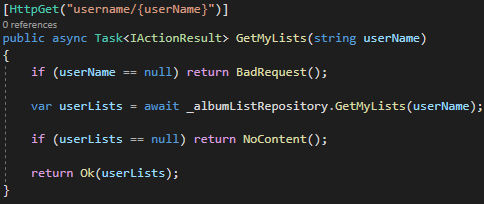
**GetPopularLists():** This GET method returns the most **popular** lists in the service. It does this by ordering the list by the *count* of the **Stars** that the list has received. For example, the list with the highest stars would be at the top of the lists through the use of this method. The **AlbumList** class contains a **Stars** data model.

**/api/AlbumList/popular/**



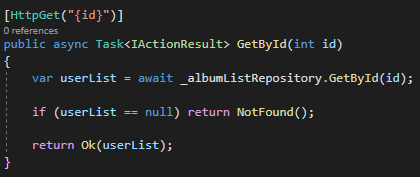
**GetMyLists():** This GET method returns the lists that have been added by the user with that specific username. This ensures that they can keep their own lists in a convenient location, and also helps identify which lists they can and cannot perform CRUD capabilities on. It does this by sorting the lists by ensuring that the username of the current device matches that of the created list. The username attribute is present in the **AlbumList** class.

**/api/AlbumList/username/{username}**



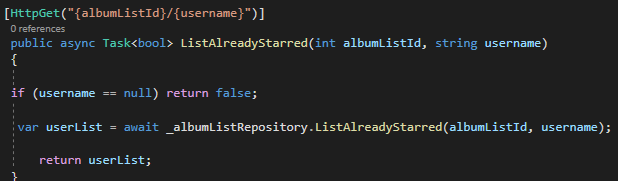
**GetById():** This GET method the returns the lists for the **ID** that is supplied by the user. It does this by taking in the ID from the user, and comparing it to the ID present in the lists. It returns the first element if found, and throws an error in there is not match.

**/api/AlbumList/{id}**



**ListAlreadyStarred():** This method also finds an **AlbumList** by it’s **ID**, and then checks if that list has already had any **stars** applied to it. If it has it returns a **true** value. Otherwise it will return a **false** value if the list has not been starred.

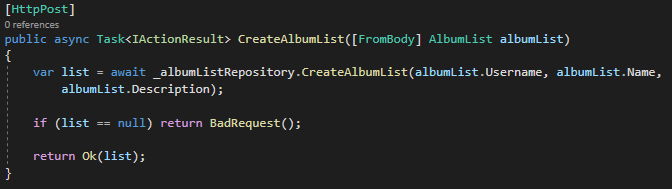
**/api/AlbumList/{id}/{username}**



**POST Methods:**

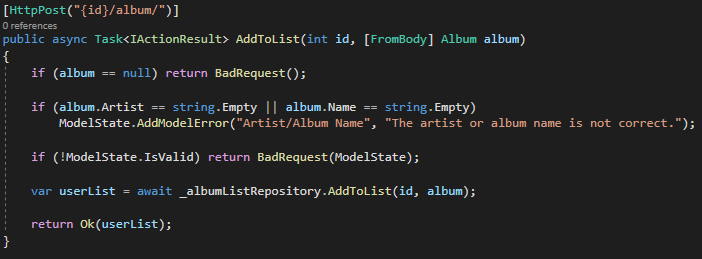
**CreateAlbumList():** This POST method allows the user to create an AlbumList. It does this using the username, the name of the list and a description. Only the name of the list and the description need to be supplied by the user for the method to work. It works by essentially creating an AlbumList object based on these parameters.

**/api/AlbumList**

****

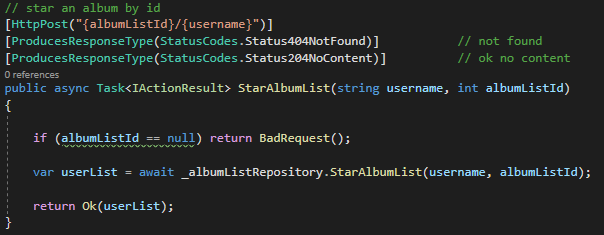
**AddToList():** This POST method allows the user to add an album to an existing AlbumList. It does this by taking in the list id, and an Album object (which is compared against the Last.fm API). If the Album is null (in the case of non-existence, for example), an error is returned. Additionally, if the Album is found successfully, it will retrieve the album art for that album automatically.

**/api/AlbumList/{id}/album**



**StarAlbumList():** This POST method allows the user to “star” an AlbumList, indicating that they wish for it to be in their favourites. It does this by taking in the username and the ID of the AlbumList and using a Star object to either add or remove a star on the list.

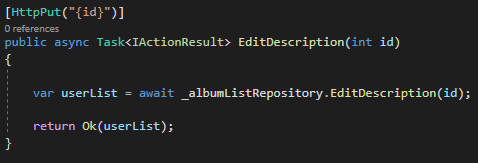
**/api/AlbumList/{id}/userName**



**PUT Methods:**

**EditDescription():** This PUT method allows the user to update the description of the AlbumList they are currently viewing. It does this by taking in the ID of the AlbumList, and updating the description of the user input to the Description attribute of the AlbumList class.

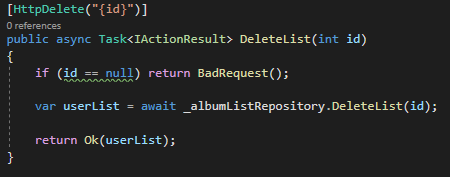
**/api/AlbumList/{id}**



**DELETE Methods:**

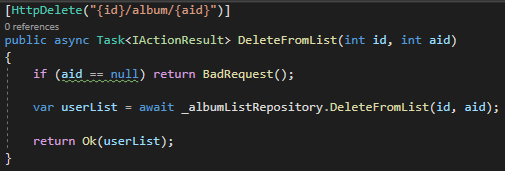
**DeleteList():** This DELETE method simply allows the user to delete an AlbumList. It used to ID to do this. If the ID is found, the list is deleted.

**/api/AlbumList/{id}**

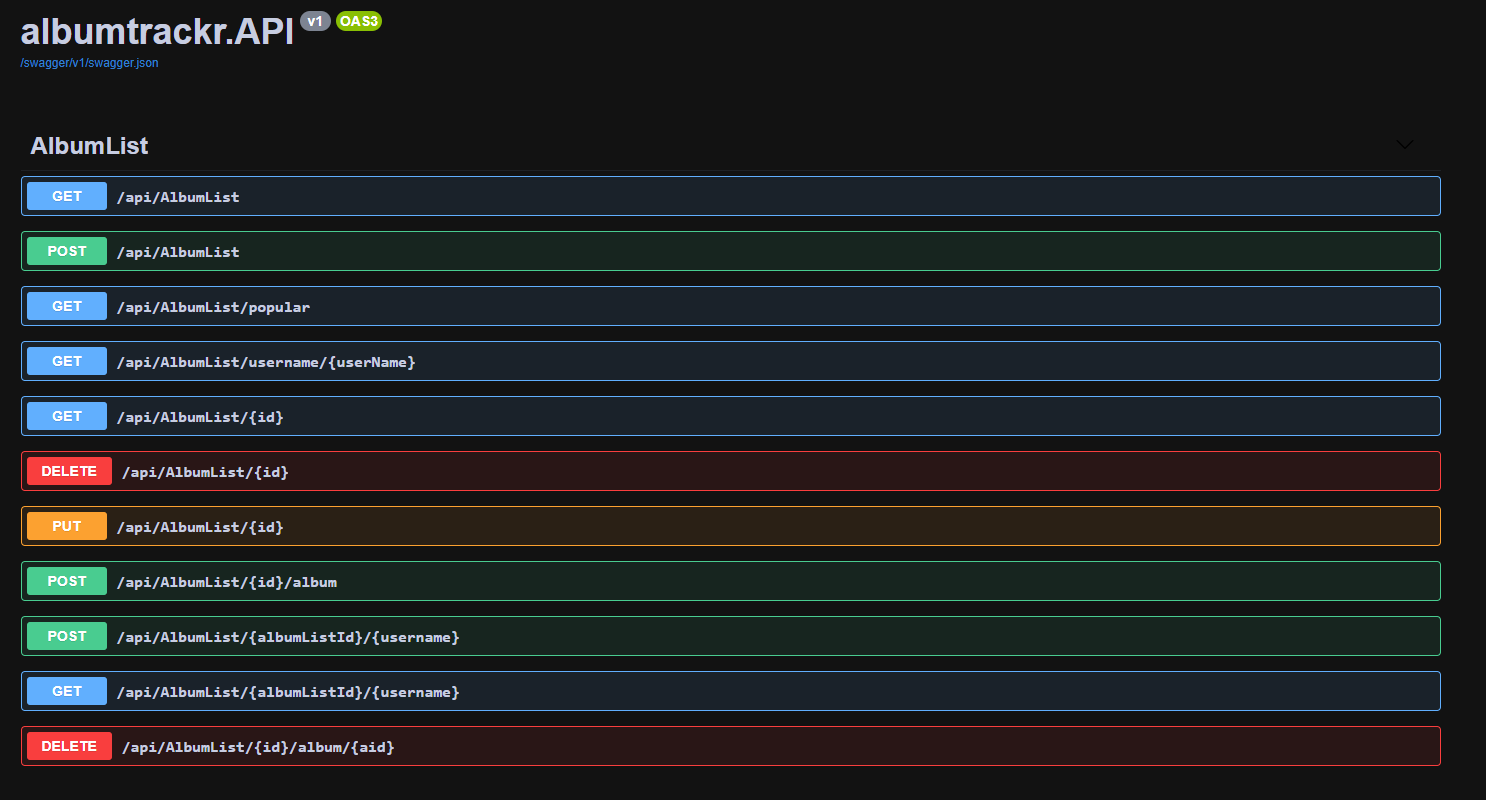


**DeleteFromList():** This DELETE method allows the user to delete an Album from an AlbumList. It uses the list ID and the Album ID to accomplish this. If the IDs match up, the album is removed from the list.

**/api/AlbumList/{id}/album/{aid}**



**Screenshot of Swagger UI Test Page**



**Deployment of Application**

The application is deployed through a pipeline on Azure. Every time we make a commit to our Github repository, the application rebuilds.

**Azure App Service Settings**

**Application Breakdown**

This section will contain screenshots of the applications features in use, a testing report, the process of internationalisation etc

**Miscellaneous**

This section will contain some code quality analysis.

**Conclusion**

In this section, we will talk about what we have learned through the development of this application, and whether or not we achieved the goals we have set ourselves.