**TestFacebookPostResource**

|  |  |
| --- | --- |
| ID | **Test 1 - testPost()** |
| Description | Test to detect errors on the implementation of post’s sharing on Facebook using RESTful services. (I) |
| Input | The *FacebookPostResource* class is used in order to invoke the service using the URI *https://graph.facebook.com/me/feed?****access\_token****=&****message****=* from our application.   * **access\_token** = personal access token of the actual user. * **message** = “This is a JUnit test.” |
| Expected Output | We expect the method in this test to return *true*, which will mean that the post was published successfully. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 2 - testPostNull()** |
| Description | Test to detect errors on the implementation of post’s sharing on Facebook using RESTful services. (II) |
| Input | The *FacebookPostResource* class is used in order to invoke the service using the URI *https://graph.facebook.com/me/feed?****access\_token****=&****message****=* from our application.   * **access\_token** = personal access token of the actual user. * **message** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

**TestFacebookResource**

|  |  |
| --- | --- |
| ID | **Test 3 - testGetPostsEmpty()** |
| Description | Test to detect errors on the implementation of getting posts from a Facebook page using RESTful services. (I) |
| Input | The *FacebookResource* class is used in order to invoke the service using the URI  *https://graph.facebook.com/v2.9/****\*****/posts?****access\_token****=* from our application.   * **\*** = “”. * **access\_token** = personal access token of the actual user. |
| Expected Output | We expect the method to return *null* as there is no page with an empty ID. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 4 - testGetPostsNull()** |
| Description | Test to detect errors on the implementation of getting posts from a Facebook page using RESTful services. (II) |
| Input | The *FacebookResource* class is used in order to invoke the service using the URI  *https://graph.facebook.com/v2.9/****\*****/posts?****access\_token****=* from our application.   * **\*** = *null*. * **access\_token** = personal access token of the actual user. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 5 - testSearchPagesEmpty()** |
| Description | Test to detect errors on the implementation of searching pages on Facebook using RESTful services. (I) |
| Input | The *FacebookResource* class is used in order to invoke the service using the URI  *https://graph.facebook.com/v2.9/search?****access\_token****=&****q****=&type=page* from our application.   * **access\_token** = personal access token of the actual user. * **q** = “”. |
| Expected Output | We expect the method to return *null* as there is no page with an empty ID. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 6 - testSearchPagesFound()** |
| Description | Test to detect errors on the implementation of searching pages on Facebook using RESTful services. (II) |
| Input | The *FacebookResource* class is used in order to invoke the service using the URI  *https://graph.facebook.com/v2.9/search?****access\_token****=&****q****=&type=page* from our application.   * **access\_token** = personal access token of the actual user. * **q** = “Lady+Gaga”. |
| Expected Output | We expect the method not to return *null* as there exists pages related to “Lady Gaga”.  Also, to return data, the size of which is bigger than 0. This means there is at least one item returned.  Also, to return a page in the first place, the name of which is “Lady Gaga”. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 7 - testSearchPagesNull()** |
| Description | Test to detect errors on the implementation of searching pages on Facebook using RESTful services. (III) |
| Input | The *FacebookResource* class is used in order to invoke the service using the URI  *https://graph.facebook.com/v2.9/search?****access\_token****=&****q****=&type=page* from our application.   * **access\_token** = personal access token of the actual user. * **q** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

**TestSpotifyResource**

|  |  |
| --- | --- |
| ID | **Test 24 - testGetAlbumEmpty()** |
| Description | Test to detect errors on the implementation of getting albums from Spotify using RESTful services. (I) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/albums/****\*****?market=ES* from our application.   * **\*** = “”. |
| Expected Output | We expect the method to return *null* as there is no album with an empty ID. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 25 - testGetAlbumFound()** |
| Description | Test to detect errors on the implementation of getting albums from Spotify using RESTful services. (II) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/albums/****\*****?market=ES* from our application.   * **\*** = “4yHr095BMG5I3IRH4ToE5l”. |
| Expected Output | We expect the method not to return *null* as there exists an album that has the ID “4yHr095BMG5I3IRH4ToE5l”.  Also, the artist of this album to be “Lady Gaga”. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 26 - testGetAlbumNull()** |
| Description | Test to detect errors on the implementation of getting albums from Spotify using RESTful services. (III) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/albums/****\*****?market=ES* from our application.   * **\*** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 27 - testGetArtistAlbumsEmpty()** |
| Description | Test to detect errors on the implementation of getting an artist’s albums from Spotify using RESTful services. (I) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/artists/****\*****/albums?album\_type=single,album,appears\_on&market=ES* from our application.   * **\*** = *“”*. |
| Expected Output | We expect the method to return *null* as there is no artist with an empty ID. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 28 - testGetArtistAlbumsFound()** |
| Description | Test to detect errors on the implementation of getting an artist’s albums from Spotify using RESTful services. (II) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/artists/****\*****/albums?album\_type=single,album,appears\_on&market=ES* from our application.   * **\*** = *“1HY2Jd0NmPuamShAr6KMms”*. |
| Expected Output | We expect the method not to return *null* as there exists an artist that has the ID “*1HY2Jd0NmPuamShAr6KMms*”.  Also, the artist must have an album called “ARTPOP”. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 29 - testGetArtistAlbumsNull()** |
| Description | Test to detect errors on the implementation of getting an artist’s albums from Spotify using RESTful services. (III) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/artists/****\*****/albums?album\_type=single,album,appears\_on&market=ES* from our application.   * **\*** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 30 - testGetArtistEmpty()** |
| Description | Test to detect errors on the implementation of getting an artist from Spotify using RESTful services. (I) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/artists/****\****from our application.   * **\*** = *“”*. |
| Expected Output | We expect the method to return *null* as there is no artist with an empty ID. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 31 - testGetArtistFound()** |
| Description | Test to detect errors on the implementation of getting an artist from Spotify using RESTful services. (II) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/artists/****\****from our application.   * **\*** = *“5WUlDfRSoLAfcVSX1WnrxN”*. |
| Expected Output | We expect the method not to return *null* as there exists an artist that has the ID “*5WUlDfRSoLAfcVSX1WnrxN*”.  Also, the artist to have 'pop' as its least important genre. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 32 - testGetArtistNull()** |
| Description | Test to detect errors on the implementation of getting an artist from Spotify using RESTful services. (III) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/artists/****\****from our application.   * **\*** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 33 - testGetTrackEmpty()** |
| Description | Test to detect errors on the implementation of getting a track from Spotify using RESTful services. (I) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/tracks/****\****from our application.   * **\*** = “”. |
| Expected Output | We expect the method to return *null* as there is no track with an empty ID. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 34 - testGetTrackFound()** |
| Description | Test to detect errors on the implementation of getting a track from Spotify using RESTful services. (II) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/tracks/****\****from our application.   * **\*** = “0QkWikH5Z3U0f79T9iuF6c”. |
| Expected Output | We expect the method not to return *null* as there exists an artist that has the ID “0QkWikH5Z3U0f79T9iuF6c”.  Also, the artist of this song to be “Lady Gaga”. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 35 - testGetTrackNull()** |
| Description | Test to detect errors on the implementation of getting a track from Spotify using RESTful services. (III) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/tracks/****\****from our application.   * **\*** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 36 - testSearchAlbumsEmpty()** |
| Description | Test to detect errors on the implementation of searching albums on Spotify using RESTful services. (I) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=album* from our application.   * **q** = “”. |
| Expected Output | We expect the method to return *null* as empty isn’t a valid search query. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 37 - testSearchAlbumsFound()** |
| Description | Test to detect errors on the implementation of searching albums on Spotify using RESTful services. (II) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=album* from our application.   * **q** = “The+Fame”. |
| Expected Output | We expect the method not to return *null* as this is valid search query.  Also, to return data, the size of which is bigger than 0. This means there is at least one item returned.  Also, the artist of the most important album called “The Fame” must be “Lady Gaga”. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 38 - testSearchAlbumsNull()** |
| Description | Test to detect errors on the implementation of searching albums on Spotify using RESTful services. (III) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=album* from our application.   * **q** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 39 - testSearchArtistsEmpty()** |
| Description | Test to detect errors on the implementation of searching artists on Spotify using RESTful services. (I) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=artist* from our application.   * **q** = “”. |
| Expected Output | We expect the method to return *null* as empty isn’t a valid search query. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 40 - testSearchArtistsFound()** |
| Description | Test to detect errors on the implementation of searching artists on Spotify using RESTful services. (II) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=artist* from our application.   * **q** = “Sia”. |
| Expected Output | We expect the method not to return *null* as this is valid search query.  Also, to return data, the size of which is bigger than 0. This means there is at least one item returned.  Also, the most important artist called “Sia” to have the ID “5WUlDfRSoLAfcVSX1WnrxN”. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 41 - testSearchArtistsNull()** |
| Description | Test to detect errors on the implementation of searching artists on Spotify using RESTful services. (III) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=artist* from our application.   * **q** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 42 - testSearchTracksEmpty()** |
| Description | Test to detect errors on the implementation of searching tracks on Spotify using RESTful services. (I) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=track* from our application.   * **q** = “”. |
| Expected Output | We expect the method to return *null* as empty isn’t a valid search query. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 43 - testSearchTracksFound()** |
| Description | Test to detect errors on the implementation of searching tracks on Spotify using RESTful services. (II) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=track* from our application.   * **q** = “Judas”. |
| Expected Output | We expect the method not to return *null* as this is valid search query.  Also, to return data, the size of which is bigger than 0. This means there is at least one item returned.  Also, the album of the most important track called “Judas” must be “Born This Way”. |
| Result | **SUCCESS** |
| Automatized | Yes |

|  |  |
| --- | --- |
| ID | **Test 44 - testSearchTracksNull()** |
| Description | Test to detect errors on the implementation of searching tracks on Spotify using RESTful services. (III) |
| Input | The *SpotifyResource* class is used in order to invoke the service using the URI *https://api.spotify.com/v1/search?****q****=&type=track* from our application.   * **q** = *null*. |
| Expected Output | We expect the method to throw a *NullPointerException*. |
| Result | **SUCCESS** |
| Automatized | Yes |

**albumView**

* The cover image of the album is successfully shown.
* The album’s title and the name of the album’s artist is successfully shown.
* The tracklist of the album (tracks and duration) is successfully shown.
* The background color is the most vibrant color of the album’s cover image.
* I click on the *Home* button and it successfully redirects me to *index.html*.
* I click on *About* *us* and it successfully redirects me to *about.html*.
* I click on the artist name and it successfully redirects me to the *artistView.jsp* of that artist.
* I click on the *Add* *to* *Spotify* button and if there’s no token for the actual session, it successfully redirects me to the Spotify Login page. Once I’ve entered my user and password, the album is added to my Spotify account and it redirects me to the *albumView.jsp*.
* If I already have a Spotify access token when I arrive to this page and I click on the *Add* *to* *Spotify* button, the album is successfully added to my Spotify account.
* If I click on the *Buy* *on* *Amazon* button, an Amazon search web page is opened with the results of the search for that album.
* If I click on a track title, it successfully redirects me to the *trackView.jsp* of that track.

**artistView**

* The artist’s profile image is successfully shown.
* The cover image of all the artist’s albums and all the albums’ titles are successfully shown.
* The Wikipedia information about the artist is successfully shown.
* The list of upcoming concerts of the artist is successfully shown.
* I click on the *Home* button and it successfully redirects me to *index*.*html*.
* I click on *About* *us* and it successfully redirects me to *about*.*html*.
* I click on an album name and it successfully redirects me to the *albumView.jsp* of that album.
* I click on the button to log in Facebook and if there’s no token for the actual session, it successfully redirects me to the Facebook Login page. Once I’ve entered my user and password, it redirects me to the *artistView.jsp* and shows me the posts of the Facebook Page of the artist.
* If I already have a Facebook access token when I arrive to this page, the posts of the artist’s page on Facebook are directly shown.

**index**

* I enter a search query and click on the *search* button and it redirects me to *searchResults.jsp* of that search query.

**searchResults**

* The artists, tracks, and albums on Spotify related to the search query I entered on *index.html* are shown on three different lists.
* If there’s no results for the search query I entered on *index.html*, a picture saying so it’s shown.
* If I click on an artist name, it successfully redirects me to the *artistView.jsp* of that artist.
* If I click on a track title, it successfully redirects me to the *trackView.jsp* of that track.
* If I click on an album title, it successfully redirects me to the *albumView.jsp* of that album.

**trackView**

* The Spotify player with the track is successfully shown.
* The YouTube video of this track is successfully shown.
* The lyrics of this track are successfully shown.
* If I start playing the track and I don’t have a Spotify access token, the player only lets me play 30 seconds of the track.
* If I already have a Spotify access token when I arrive to this page and I start playing the tracks, the Spotify application (or Spotify web if I don’t have the application installed) opens and the player lets me play the whole track.
* I click on the *Add* *to* *Spotify* button and, if there’s no token for the actual session, it successfully redirects me to the Spotify Login page. Once I’ve entered my user and password, the track is added to my Spotify account and it redirects me to the *trackView.jsp* of that track.
* If I already have a Spotify access token when I arrive to this page and I click on the *Add* *to* *Spotify* button, the track is successfully added to my Spotify account.
* I click on the *Home* button and it successfully redirects me to *index*.*html*.
* I click on *About* *us* and it successfully redirects me to *about*.*html*.
* If I click on the play button of the YouTube video it successfully starts playing.
* I select some of the lyrics and click on the *Post* *on* *Facebook* button. If there’s no token for the actual session, it successfully redirects me to the Facebook Login page. Once I’ve entered my user and password, the lyrics I selected are successfully published on my Facebook account and it redirects me to the *trackView.jsp* of that track.
* If I already have a Facebook access token when I arrive to this page, I select some of the lyrics and click on the *Post* *on* *Facebook* button, the lyrics I selected are successfully published on my Facebook account and it redirects me to the *trackView.jsp* of that track.