**Assignment 8 – Composer API, part 2**

**<yourLastName>-composer-routes.js**

* Create two new operations: updateComposerById and deleteComposerById

# **updateComposerById**

**Operation: PUT**

**Path: /api/composers/:id**

|  |  |  |
| --- | --- | --- |
| **Param Type** | **Field** | **Data Type** |
| RequestParams | id | String |
| RequestBody | firstName | String |
| RequestBody | lastName | String |

|  |  |
| --- | --- |
| **Response Code** | **Message** |
| 200 | Array of composer documents |
| 401 | Invalid composerId |
| 500 | Server Exception |
| 501 | MongoDB Exception |

**Additional Programming Requirements**

* + Wrap the code in a try/catch block.
  + Query the composers collection using the findOne() function and the RequestParams id on the Composer model.
  + Add an if…else block to composer the returned value from MongoDB.
    - if (composer)
      * Update the returned composer object by using the set() function and mapping the RequestBody fields to the returned objects parameters.
      * Call the save() function on the returned composer object and return the savedComposer object.
    - if(!composer)
      * Return a status 401 and the specified error message.
  + For all other responses returned the appropriate message depending on the status code.
  + Describe the operation using the OpenAPI Specification above the request as developer comments.

# **deleteComposerById**

**Operation: DELETE**

**Path: /api/composers/:id**

|  |  |  |
| --- | --- | --- |
| **Param Type** | **Field** | **Data Type** |
| RequestParams | id | String |

|  |  |
| --- | --- |
| **Response Code** | **Message** |
| 200 | Composer document |
| 500 | Server Exception |
| 501 | MongoDB Exception |

**Additional Programming Requirements**

* + Wrap the code in a try/catch block.
  + Call the findByIdAndDelete() function on the Composer model and use the RequestParams id as the filter criteria.
  + Either return the deleted composer document or the appropriate message depending on the status code.
  + Describe the operation using the OpenAPI Specification above the request as developer comments.
* Test one of the API’s using the generated Swagger documentation at <https://localhost:3000/api-docs> and the other using SoapUI. The documentation for installing and testing an API with SoapUI is located under the SoapUI Guide. Take screenshots of the testing you did in SoapUI and from the Swagger documentation link.
* If you run into issues, refer to the courses GitHub repository and the work you completed in WEB 340.