Jose Loperena Monzón 4518381

API documentation Final assignment data processing

1. Introduction

In this document, I will explain the different methods which the API uses and what they do. I recommend having these methods open side by side as they include comments at the top which also explain briefly what they do.

2. Movie controller.

In the class movie controller, is where all of the CRUD operations occur. Due to having 3 different endpoints, means that I have done the 4 CRUD operations the same way but with the different endpoints. This means I will explain the 4 operations as they do the same.

1. Get all films:

```
@GetMapping(value = "/amazon", produces = {"application/json",
"application/xml"})
public List<AmazonMovie> findAmazonMovies() {
    return amazonMovieRepository.findAll();
}
```

This method, gets all of the films from the specific database we have chosen. This is used for the 3 endpoints. What it does is getting all of the records that the database has.

2. Returning a specific film:

```
3. @GetMapping(value = "/amazon/{id}")
   public AmazonMovie getAmazonMovie(@PathVariable("id") int id) {
       return amazonMovieRepository.findById(id).orElse(null);
   }
```

In this method we get a specific movie via the ID located in the different databases. We find the ID of a specific film. This method is done so that we can Update and Delete specific films.

3. Inserting a film:

```
@PutMapping("/amazon")
public Message insertAmazon(@RequestBody String movie) {

    boolean okSchema = movieValidator(movie, jsonSchemaAmazon);
    if (okSchema) {

        AmazonMovie m = null;
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            m = objectMapper.readValue(movie, AmazonMovie.class);

        } catch (JsonProcessingException ex) {

            return new Message(500,ex.getMessage());
        }


        m = amazonMovieRepository.save(m); //adding the movie into the
dabatabase

        if (m != null)
            return new Message(200, "OK");
        else
            return new Message(500, "Could not save movie");
    }

    return new Message(500, "Validation error");
}
```

With this method, we insert films into the database via the API. When adding a film into the database, it is validated via JSONSchema validator. If the validation has worked, then it should be inserted and show an OK message, meanwhile if there is a validation error it does not insert and it shows an error.

4. Editing films:

```
@PostMapping("/amazon")
@ResponseBody
public String editAmazon(@RequestBody AmazonMovie movie) {

    Boolean status = Boolean.FALSE;
    AmazonMovie m =
amazonMovieRepository.findById(movie.getId()).orElse(null); //adding
the movie into the database

    if (m != null) {
        m = amazonMovieRepository.save(movie);
        if (m != null)
            status = Boolean.TRUE;
```

```
    }

    return status.toString();
}
```

This method gives us a movie ID so that we can edit the film via the Visualization. If it worked then it will show a status in the JavaScript console.

5. Delete Film:

```
6.  @DeleteMapping("/amazon")
    @ResponseBody
    public String deleteAmazon(@RequestParam("id") int id) {
        AmazonMovie m =
    amazonMovieRepository.findById(id).orElse(null);
        if (m != null) {
            amazonMovieRepository.delete(m);
            return Boolean.TRUE.toString();
        }

        return Boolean.FALSE.toString();

    }
```

This will delete a film from the database via the API.

Extra:

As said on the introduction, these methods are repeated 3 times for each endpoint but do the same thing on the 3 of them. I am showing an example in 1 of them.