### Documentation Slideshow Project Overview

This project is a Java Spring Boot application designed to showcase expertise in modern Java technologies and practices. The application provides a platform for users to manage a list of image URLs and play a slideshow with dynamic transitions. It supports the following key features:

* **Image Management**: Users can upload, remove, and search for images based on specific parameters such as keywords or duration.
* **Slideshow Creation**: Users can create and manage slideshows, associating images with specific display durations and ordering.
* **Proof of Play**: Every time an image is replaced in the slideshow, a proof-of-play event is triggered to track the transitions between images.
* **Scalable and Robust API**: The application includes a well-structured API that enables the integration of additional functionality such as the modification of slideshow orders or management of images.

### Server Response Structure

Each response from the server will have a standard JSON structure. Here are the main fields in the response:

* **data**: This field contains the useful data returned in case of a successful operation or a detailed error description in case of failure.
* **status**: This field indicates the status of the operation. It can have one of two values:
  + **"success"** — The operation was successfully completed.
  + **"error"** — There was an error during the execution of the operation.
* **code**: This is the numeric status code that defines the result of the request. You can refer to the /statusCodes route to see all possible status codes.
* **message**: A message explaining the result of the operation. This message will indicate if the operation was successful or provide details about the error.

#### Example of a successful response:

{

    "data": [],

    "status": "success",

    "code": 201,

    "message": "Successfully processed operation"

}

Example of an error response:

{

    "data": [],

    "status": "error",

    "code": 6001,

    "message": "Image not found"

}

### 1. ****Get Status Codes****

This endpoint provides a list of all possible status codes and their meanings used throughout the API. This is helpful for understanding the meaning behind each response code returned by the API.

* **Service**: general
* **Method**: GET
* **URL**: /statusCodes

#### Request URL Example:

http://localhost:8080/statusCodes

#### Response:

* **Code**: 200 OK
* **Headers**:
  + Content-Type: application/json
  + Content-Length: 539
* **Body Example**:

{

    "200": "Ok",

    "201": "Successfully processed operation",

    "1000": "Failed validation",

    "1001": "Failed validation image type or content",

    "1002": "Failed validation duration of image",

    "2000": "Slideshow Not Found",

    "3000": "Validation Failed",

    "4000": "Already exists",

    "4001": "Invalid request body format",

    "4002": "Invalid request parameters format",

    "5000": "Internal Server Error",

    "5001": "Database Operation Failed",

    "5002": "Error loading file",

    "6001": "Image not found",

    "6002": "Error occurred while mapping data",

    "6003": "JSON serialization error occurred"

}

### 2. ****Add Image****

This endpoint allows users to add an image to the system, along with the display duration for a slideshow.

* **Service**: slideshow
* **Method**: POST
* **URL**: http://localhost:8080/api/v1/addImage

#### Headers:

* Content-Type: application/json; charset=utf-8

#### Request Body Example:

{

    "url": "https://fastly.picsum.photos/id/643/200/300.jpg?hmac=rS-MHa0BIMHdAgm-FZ7QM36aRKEAzzhSRNRv5n4uqGc",

    "duration": 30

}

#### Parameters:

* **url**: The URL of the image to be added (must be a valid image URL).
* **duration**: The duration (in seconds) that the image will be displayed in the slideshow.

#### Response:

* **Code**: 201 Created
* **Body**:

{

    "data": [

        {

            "imageId": 26

        }

    ],

    "status": "success",

    "code": 201,

    "message": "Successfully processed operation"

}

### 3. ****Remove Image****

This endpoint removes an image from the system by its ID.

* **Service**: slideshow
* **Method**: DELETE
* **URL**: http://localhost:8080/api/v1/deleteImage/{imageId}

#### Request URL Example:

http://localhost:8080/api/v1/deleteImage/6

#### Parameters:

* **imageId**: The ID of the image to be deleted.

#### Response:

* **Code**: 201 Created
* **Body**:

{

    "data": [],

    "status": "success",

    "code": 201,

    "message": "Successfully processed operation"

}

### 4. ****Add Slideshow****

This endpoint allows users to create a new slideshow with a list of images and their respective display durations.

* **Service**: slideshow
* **Method**: POST
* **URL**: http://localhost:8080/api/v1/addSlideshow

#### Headers:

* Content-Type: application/json; charset=utf-8

#### Request Body Example:

{

    "name": "test2",

    "images": [

        {

            "image\_id": 10,

            "duration": 10

        },

        {

            "image\_id": 5,

            "duration": -1

        }

    ]

}

Parameters:

* **name**: The name of the slideshow (optional).
* **images**: An array of image objects, where each object contains:
  + **image\_id**: The ID of the image.
  + **duration**: The display duration for the image in the slideshow.

#### Response:

* **Code**: 201 Created
* **Body**:

{

    "data": [],

    "status": "success",

    "code": 201,

    "message": "Successfully processed operation"

}

### 5. ****Delete Slideshow****

This endpoint allows users to remove a slideshow by its ID.

* **Service**: slideshow
* **Method**: DELETE
* **URL**: http://localhost:8080/api/v1/deleteSlideshow/{slideshowId}

#### Request URL Example:

http://localhost:8080/api/v1/deleteSlideshow/6

#### Parameters:

* **slideshowId**: The ID of the slideshow to be deleted.

#### Response:

* **Code**: 201 Created
* **Body**:

{

    "data": [],

    "status": "success",

    "code": 201,

    "message": "Successfully processed operation"

}

### 6. ****Search Image****

This endpoint allows users to search for images based on specific keywords or duration.

* **Service**: slideshow
* **Method**: GET
* **URL**: http://localhost:8080/api/v1/images/search

#### Headers:

* Content-Type: application/json; charset=utf-8

#### Request Body Example:

{

    "keyword": "test",

    "duration": 30

}

#### Parameters:

* **keyword**: A search keyword to find images based on URL or other associated data.
* **duration**: The duration in seconds for filtering images.

#### Response:

* **Code**: 200 OK
* **Body**:

{

    "data": [

        {

            "image": {

                "id": 10,

                "url": "https://fastly.picsum.photos/id/1080/200/300.jpg?hmac=3jCG4X8ni9cpiEppj6KVvqhpJ\_owIFtDQOABJmhG1I0",

                "duration": 30,

                "type": "image/jpeg",

                "addedTime": "2024-12-17T01:36:11.399025"

            },

            "slideshow": [

                {

                    "id": 4,

                    "slideshowId": 5,

                    "imageId": 10,

                    "duration": 10

                },

                {

                    "id": 8,

                    "slideshowId": 7,

                    "imageId": 10,

                    "duration": 30

                },

                {

                    "id": 11,

                    "slideshowId": 8,

                    "imageId": 10,

                    "duration": 30

                }

            ]

        }

    ],

    "status": "success",

    "code": 200,

    "message": "Ok"

}

### 7. ****Get Slideshow Order****

This endpoint allows users to retrieve the images of a slideshow ordered by their addition time, ensuring the correct sequence during playback.

* **Service**: slideshow
* **Method**: GET
* **URL**: /api/{vers}/slideShow/{slideshowId}/slideshowOrder

#### Parameters:

* **vers**: The version of the API (e.g., v1).
* **slideshowId**: The ID of the slideshow whose images are being retrieved.

#### Request URL Example:

http://localhost:8080/api/v1/slideShow/5/slideshowOrder

#### Response:

* **Code**: 200 OK
* **Headers**:
  + Content-Type: text/event-stream;charset=UTF-8
  + Transfer-Encoding: chunked
* **Body Example**:

data:{"data":[

    {

        "imageData": {

            "id": 5,

            "url": "https://fastly.picsum.photos/id/892/200/300.jpg?hmac=9MUtm-RM2UIFVmP8I80S9TuWKk93ZEPqQLpOf-y1BwE",

            "duration": 6,

            "type": "image/jpeg",

            "addedTime": "2024-12-15T01:20:09.528432"

        },

        "imageId": 5,

        "slideshowId": 5,

        "duration": 2

    }

],"status":"success","code":200,"message":"Ok"}

data:{"data":[

    {

        "imageData": {

            "id": 10,

            "url": "https://fastly.picsum.photos/id/1080/200/300.jpg?hmac=3jCG4X8ni9cpiEppj6KVvqhpJ\_owIFtDQOABJmhG1I0",

            "duration": 30,

            "type": "image/jpeg",

            "addedTime": "2024-12-17T01:36:11.399025"

        },

        "imageId": 10,

        "slideshowId": 5,

        "duration": 10

    }

],"status":"success","code":200,"message":"Ok"}

### 8. ****Proof of Play****

This endpoint allows the system to record an event whenever an image is replaced by the next one in a slideshow. This helps track image transitions and monitor playback status.

* **Service**: slideshow
* **Method**: POST
* **URL**: /api/{vers}/slideShow/{slideshowId}/proof-of-play/{imageId}

#### Parameters:

* **vers**: The version of the API (e.g., v1).
* **slideshowId**: The ID of the slideshow.
* **imageId**: The ID of the image whose transition is being recorded.

#### Request URL Example:

http://localhost:8080/api/v1/slideShow/5/proof-of-play/10

#### Response:

* **Code**: 201 Created
* **Headers**:
  + Content-Type: application/json
  + Content-Length: 86
* **Body Example**:

{

    "data": [],

    "status": "success",

    "code": 201,

    "message": "Successfully processed operation"

}

### Error Descriptions

#### Error 1: Attempt to Add an Already Existing Image

* **Situation**: A request is sent to add an image that has already been added.
* **Error**: The changes cannot be applied, and the system returns an error.

**Request Example**:

{

    "url": "https://fastly.picsum.photos/id/643/200/300.jpg?hmac=rS-MHa0BIMHdAgm-FZ7QM36aRKEAzzhSRNRv5n4uqGc",

    "duration": 30

}

**Response**:

{

    "data": [],

    "status": "error",

    "code": 4000,

    "message": "Already exists"

}

#### Error 2: Attempt to Add an Image with Invalid Data

* **Situation**: The request contains invalid data.
* **Error**: The changes cannot be applied, and the system indicates invalid body data.

**Request Example**:

{

    "url": "https://fastly.picsum.photos/id/643/200/300.jpg?hmac=rS-MHa0BIMHdAgm-FZ7QM36aRKEAzzhSRNRv5n4uqGc",

    "some\_test":

}

**Response**:

{

    "data": [],

    "status": "error",

    "code": 4001,

    "message": "Invalid request body format"

}

#### Error 3: Attempt to Delete a Non-Existent Image

* **Situation**: A request is sent to delete an image that does not exist.
* **Error**: The changes cannot be applied, and the system indicates the image was not found.

**Request Example**:

http://localhost:8080/api/v1/deleteImage/6

**Response**:

{

    "data": [],

    "status": "error",

    "code": 6001,

    "message": "Image not found"

}