Sure! Below is a detailed documentation for creating a Gallery API using Spring Boot for the backend and Angular for the frontend. This documentation will cover the following:

1. Introduction

2. Backend (Spring Boot)

- Project Setup

- Dependencies

- Directory Structure

- Configuration (application.properties)

- Entity, Repository, Service, Controller

- Handling CORS and File Upload

- Testing the API with Postman

3. Frontend (Angular)

- Project Setup

- Dependencies

- Directory Structure

- Service, Component

- Handling File Upload and Displaying Progress

- Styling with CSS

- Error Handling and UX

4. Conclusion

---

1. Introduction

This project involves creating a full-stack gallery application where users can upload images to a gallery, view them, and delete them. The backend is developed using Spring Boot with MySQL as the database, while the frontend is built using Angular. Cloudinary is used to store the images.

---

2. Backend (Spring Boot)

2.1 Project Setup

1. Create a Spring Boot project using Spring Initializr (https://start.spring.io/).

2. Choose the following dependencies:

- Spring Web

- Spring Data JPA

- MySQL Driver

- Spring Boot DevTools

- Cloudinary SDK (if not available, use Maven Central to add manually)

2.2 Dependencies

Add the following dependencies to your `pom.xml`:

```xml

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Boot Starter Data JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- MySQL Driver -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Cloudinary SDK -->

<dependency>

<groupId>com.cloudinary</groupId>

<artifactId>cloudinary</artifactId>

<version>1.33.0</version>

</dependency>

<!-- Spring Boot DevTools -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter Test -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

```

2.3 Directory Structure

```

src/

|-- main/

| |-- java/

| | |-- com/

| | | |-- example/

| | | | |-- gallery/

| | | | | |-- controller/

| | | | | |-- entity/

| | | | | |-- repository/

| | | | | |-- service/

| | | | | | |-- impl/

|-- resources/

| |-- application.properties

```

2.4 Configuration (`application.properties`)

Configure the MySQL database and Cloudinary settings in `application.properties`:

```properties

MySQL Configuration

spring.datasource.url=jdbc:mysql://localhost:3306/gallerydb

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

Cloudinary Configuration

cloudinary.cloud-name=your\_cloud\_name

cloudinary.api-key=your\_api\_key

cloudinary.api-secret=your\_api\_secret

CORS Configuration

spring.mvc.cors.allowed-origins=http://localhost:4200

spring.mvc.cors.allowed-methods=GET,POST,DELETE

spring.mvc.cors.allowed-headers=

Server Port

server.port=8081

```

2.5 Entity, Repository, Service, Controller

1. Entity Class:

```java

package com.example.gallery.entity;

import javax.persistence. ;

@Entity

public class Image {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String publicId;

private String url;

// Getters and Setters

}

```

2. Repository Interface:

```java

package com.example.gallery.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.gallery.entity.Image;

public interface ImageRepository extends JpaRepository<Image, Long> {

}

```

3. Service Interface:

```java

package com.example.gallery.service;

import com.example.gallery.entity.Image;

import org.springframework.web.multipart.MultipartFile;

import java.util.List;

public interface ImageService {

Image uploadImage(MultipartFile file);

List<Image> getAllImages();

void deleteImage(String publicId);

}

```

4. Service Implementation:

```java

package com.example.gallery.service.impl;

import com.cloudinary.Cloudinary;

import com.cloudinary.utils.ObjectUtils;

import com.example.gallery.entity.Image;

import com.example.gallery.repository.ImageRepository;

import com.example.gallery.service.ImageService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

import java.util.List;

import java.util.Map;

@Service

public class ImageServiceImpl implements ImageService {

@Autowired

private Cloudinary cloudinary;

@Autowired

private ImageRepository imageRepository;

@Override

public Image uploadImage(MultipartFile file) {

try {

Map uploadResult = cloudinary.uploader().upload(file.getBytes(), ObjectUtils.emptyMap());

Image image = new Image();

image.setPublicId((String) uploadResult.get("public\_id"));

image.setUrl((String) uploadResult.get("url"));

return imageRepository.save(image);

} catch (IOException e) {

e.printStackTrace();

return null;

}

}

@Override

public List<Image> getAllImages() {

return imageRepository.findAll();

}

@Override

public void deleteImage(String publicId) {

try {

cloudinary.uploader().destroy(publicId, ObjectUtils.emptyMap());

Image image = imageRepository.findByPublicId(publicId);

imageRepository.delete(image);

} catch (IOException e) {

e.printStackTrace();

}

}

}

```

5. Controller:

```java

package com.example.gallery.controller;

import com.example.gallery.entity.Image;

import com.example.gallery.service.ImageService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation. ;

import org.springframework.web.multipart.MultipartFile;

import java.util.List;

@RestController

@RequestMapping("/api/gallery")

public class ImageController {

@Autowired

private ImageService imageService;

@PostMapping("/upload")

public ResponseEntity<Image> uploadImage(@RequestParam("file") MultipartFile file) {

Image image = imageService.uploadImage(file);

return ResponseEntity.ok(image);

}

@GetMapping("/all")

public ResponseEntity<List<Image>> getAllImages() {

List<Image> images = imageService.getAllImages();

return ResponseEntity.ok(images);

}

@DeleteMapping("/delete/{publicId}")

public ResponseEntity<Void> deleteImage(@PathVariable String publicId) {

imageService.deleteImage(publicId);

return ResponseEntity.noContent().build();

}

}

```

2.6 Handling CORS and File Upload

- CORS: Allow cross-origin requests by configuring CORS in the `application.properties` file or using a `@CrossOrigin` annotation in the controller.

- File Upload: Handle file uploads using the `MultipartFile` class, and store them in Cloudinary.

2.7 Testing the API with Postman

1. Upload an Image:

- URL: `POST http://localhost:8081/api/gallery/upload`

- Body: Form-data with a key `file` and select the file to upload.

2. Get All Images:

- URL: `GET http://localhost:8081/api/gallery/all`

3. Delete an Image:

- URL: `DELETE http://localhost:8081/api/gallery/delete/{publicId}`

---

3. Frontend (Angular)

3.1 Project Setup

1. Create an Angular project using Angular CLI.

2. Install necessary dependencies:

- `@angular/common`

- `@angular/http-client`

3.2 Directory Structure

```

src/

|-- app/

| |-- components/

| | |-- gallery/

| | | |-- gallery.component.ts

| | | |-- gallery.component.html

| | | |-- gallery.component.css

| |-- services/

| | |-- gallery.service.ts

|-- assets/

|-- environments/

```

3.3 Service and Component

1. Gallery Service:

```typescript

import { Injectable } from '@angular/core';

import { HttpClient, HttpRequest, HttpEventType, HttpResponse } from '@angular/common/http';

import { Observable } from 'rxjs';

import { Image } from '../models/image.model';

@Injectable({

providedIn:

'root'

})

export class GalleryService {

private apiUrl = 'http://localhost:8081/api/gallery';

constructor(private http: HttpClient) { }

uploadImage(file: File): Observable<any> {

const formData: FormData = new FormData();

formData.append('file', file);

const req = new HttpRequest('POST', `${this.apiUrl}/upload`, formData, {

reportProgress: true,

});

return this.http.request(req);

}

getAllImages(): Observable<Image[]> {

return this.http.get<Image[]>(`${this.apiUrl}/all`);

}

deleteImage(publicId: string): Observable<void> {

return this.http.delete<void>(`${this.apiUrl}/delete/${publicId}`);

}

}

```

2. Gallery Component:

```typescript

import { Component, OnInit } from '@angular/core';

import { GalleryService } from '../../services/gallery.service';

import { Image } from '../../models/image.model';

@Component({

selector: 'app-gallery',

templateUrl: './gallery.component.html',

styleUrls: ['./gallery.component.css']

})

export class GalleryComponent implements OnInit {

images: Image[] = [];

selectedFile: File | null = null;

uploadProgress: number = 0;

estimatedTime: number = 0;

uploading: boolean = false;

startTime: number = 0;

constructor(private galleryService: GalleryService) { }

ngOnInit(): void {

this.loadImages();

}

onFileSelected(event: any): void {

this.selectedFile = event.target.files[0];

}

uploadImage(): void {

if (this.selectedFile) {

this.uploading = true;

this.startTime = Date.now();

this.galleryService.uploadImage(this.selectedFile).subscribe(event => {

if (event.type === HttpEventType.UploadProgress) {

if (event.total) {

this.uploadProgress = Math.round((100 event.loaded) / event.total);

const timeElapsed = (Date.now() - this.startTime) / 1000; // time in seconds

const uploadSpeed = event.loaded / timeElapsed; // bytes per second

const remainingBytes = event.total - event.loaded;

this.estimatedTime = Math.round(remainingBytes / uploadSpeed); // remaining time in seconds

}

} else if (event instanceof HttpResponse) {

this.uploading = false;

this.loadImages();

}

}, error => {

this.uploading = false;

console.error('Upload failed', error);

});

}

}

deleteImage(publicId: string): void {

this.galleryService.deleteImage(publicId).subscribe(response => {

this.loadImages();

});

}

loadImages(): void {

this.galleryService.getAllImages().subscribe(data => {

this.images = data;

});

}

}

```

3. HTML Template:

Refer to the HTML template provided earlier in the conversation.

3.4 Styling with CSS

Refer to the CSS provided earlier in the conversation for basic styling.

3.5 Error Handling and UX

Implement appropriate error handling in the Angular service and component to improve user experience, such as displaying alerts or notifications when the upload fails or when an image is deleted successfully.

---

4. Conclusion

This document provides a comprehensive guide to building a gallery application using Spring Boot and Angular. It includes setting up the backend with image upload and retrieval functionality, as well as the frontend with file upload progress display. This application can be expanded with additional features like user authentication, image categorization, or adding metadata to images.