**Sub-task 1: Resource Service**

Project structure:  
A screenshot of a computer program

Description automatically generated

Controller layer:

@Controller  
@RequestMapping(path = "/resources")  
@RequiredArgsConstructor  
public class ResourceController {  
  
 private final ResourceService resourceService;  
  
 @PostMapping(consumes = {MediaType.*MULTIPART\_FORM\_DATA\_VALUE*})  
 public ResponseEntity<IdResponse> saveAudio(@RequestPart MultipartFile mp3File) {  
 var id = resourceService.saveAudio(mp3File);  
 var response = new IdResponse(id);  
 return ResponseEntity.*ok*(response);  
 }  
  
 @GetMapping(path = "/{id}")  
 public ResponseEntity<byte[]> getAudio(@PathVariable Long id, ServletContext servletContext) {  
 var audioData = resourceService.getAudio(id);  
 return ResponseEntity.*ok*(audioData);  
 }  
  
 @DeleteMapping  
 public ResponseEntity<Set<Long>> deleteAudioByIds(@RequestParam(name = "id") String idsSeparatedByComma) {  
 var ids = Arrays.*stream*(idsSeparatedByComma.split(","))  
 .map(String::trim)  
 .map(Long::*valueOf*)  
 .collect(Collectors.*toSet*());  
 resourceService.deleteAudio(ids);  
 return ResponseEntity.*ok*(ids);  
 }  
}

Service layer:

@Service  
@Slf4j  
@RequiredArgsConstructor  
public class ResourceServiceImpl implements ResourceService {  
  
 private final ResourceRepository resourceRepository;  
  
 private final SongServiceClient songServiceClient;  
  
 @Override  
 @Transactional  
 public Long saveAudio(MultipartFile mp3File) {  
 try {  
 var audio = resourceRepository.save(Audio.*builder*().audioData(mp3File.getBytes()).build());  
  
 //detecting the file type  
 var handler = new BodyContentHandler();  
 var metadata = new Metadata();  
 var inputStream = mp3File.getInputStream();  
 var parseContext = new ParseContext();  
  
 //Mp3 parser  
 var mp3Parser = new Mp3Parser();  
 mp3Parser.parse(inputStream, handler, metadata, parseContext);  
 var lyrics = new LyricsHandler(inputStream, handler);  
  
 while (lyrics.hasLyrics()) {  
 *log*.debug(lyrics.toString());  
 }  
  
 *log*.debug("Contents of the document: {}", handler);  
 *log*.debug("Metadata of the document:");  
 String[] metadataNames = metadata.names();  
  
 for (String name : metadataNames) {  
 *log*.debug(name + ": " + metadata.get(name));  
 }  
 var songMetadataDto = SongMetadataDto.*builder*().name(metadata.get("dc:title"))  
 .artist(metadata.get("xmpDM:artist"))  
 .album(metadata.get("album"))  
 .length(metadata.get("xmpDM:duration"))  
 .resourceId(audio.getId())  
 .year(metadata.get("year"))  
 .build();  
 songServiceClient.saveMetadata(songMetadataDto);  
 return audio.getId();  
 } catch (Exception e) {  
 *log*.error(e.getMessage());  
 throw new ResourceServiceException("Could not save audio. " + e.getMessage());  
 }  
 }  
  
 @Override  
 @Transactional(readOnly = true)  
 public byte[] getAudio(long id) {  
 var audio = resourceRepository.findById(id)  
 .orElseThrow(() -> new ResourceNotFoundException(String.*format*("Audio file with ID %d is not found", id)));  
 return audio.getAudioData();  
 }  
  
 @Override  
 @Transactional  
 public void deleteAudio(Set<Long> ids) {  
 resourceRepository.deleteAllById(ids);  
 }

Entity:

A screen shot of a computer program

Description automatically generated

Dtos:

@Builder  
public record SongMetadataDto(String name, String artist, String album, String length, Long resourceId, String year) {  
  
}

public record IdResponse(Long id) {  
}

Application.yaml:  
  
A screen shot of a computer

Description automatically generated

**Sub-task 2: Song Service**

Project structure:  
A screenshot of a computer

Description automatically generated

Controller layer:

@Controller  
@RequestMapping(path = "/songs")  
@RequiredArgsConstructor  
public class SongController {  
  
 private final SongService songService;  
  
 @PostMapping  
 public ResponseEntity<IdResponse> saveSongMetadata(@RequestBody SongMetadataDto songMetadataDto) {  
 var songMetadata = songMetadataDtoToSongMetadata(songMetadataDto);  
 var id = songService.saveSongMetadata(songMetadata);  
 var response = new IdResponse(id);  
 return ResponseEntity.*ok*(response);  
 }  
  
 @GetMapping(path = "/{id}")  
 public ResponseEntity<SongMetadataDto> getSongMetadata(@PathVariable Long id) {  
 var songMetaData = songService.getSongMetadata(id);  
 var songMetadataDto = songMetadataToSongMetadataDto(songMetaData);  
 return ResponseEntity.*ok*(songMetadataDto);  
 }  
  
 @DeleteMapping(path = "/{id}")  
 public ResponseEntity<Set<Long>> deleteSongMetadata(String idsSeparatedByComma) {  
 var ids = Arrays.*stream*(idsSeparatedByComma.split(","))  
 .map(String::trim)  
 .map(Long::*valueOf*)  
 .collect(Collectors.*toSet*());  
 songService.deleteSongMetadata(ids);  
 return ResponseEntity.*ok*(ids);  
 }  
  
 private SongMetadata songMetadataDtoToSongMetadata(SongMetadataDto songMetadataDto) {  
 return SongMetadata.*builder*()  
 .name(songMetadataDto.name())  
 .artist(songMetadataDto.artist())  
 .album(songMetadataDto.album())  
 .length(songMetadataDto.length())  
 .resourceId(songMetadataDto.resourceId())  
 .year(songMetadataDto.year())  
 .build();  
 }  
  
 private SongMetadataDto songMetadataToSongMetadataDto(SongMetadata songMetaData) {  
 return SongMetadataDto.*builder*()  
 .name(songMetaData.getName())  
 .artist(songMetaData.getArtist())  
 .album(songMetaData.getAlbum())  
 .length(songMetaData.getLength())  
 .resourceId(songMetaData.getResourceId())  
 .year(songMetaData.getYear())  
 .build();  
 }  
}

Service layer:

@Service  
@Slf4j  
@RequiredArgsConstructor  
public class SongServiceImpl implements SongService {  
  
 private final SongRepository songRepository;  
  
  
 @Override  
 @Transactional(readOnly = true)  
 public SongMetadata getSongMetadata(long id) {  
 var songMetadata = songRepository.findById(id)  
 .orElseThrow(() -> new ResourceNotFoundException(String.*format*("Song Metadata with ID %d is not found", id)));  
 return songMetadata;  
 }  
  
 @Override  
 @Transactional  
 public Long saveSongMetadata(SongMetadata songMetadata) {  
 songRepository.save(songMetadata);  
 return songMetadata.getId();  
 }  
  
 @Override  
 @Transactional  
 public void deleteSongMetadata(Set<Long> ids) {  
 songRepository.deleteAllById(ids);  
 }  
}

Entity:  
A screen shot of a computer program

Description automatically generated

Application.yaml  
  
A screen shot of a computer

Description automatically generated