Desenvolvimento de uma Aplicação de Gravação e Transcrição de Áudio com Spring Boot: Arquitetura e Boas Práticas

Introdução

Este artigo descreve detalhadamente a implementação de uma solução baseada em **Spring Boot** para gravação, conversão e transcrição de áudio utilizando a **Google Cloud Speech API**. A aplicação foi desenvolvida com uma arquitetura modular e de alta escalabilidade, priorizando boas práticas de engenharia de software e padrões de projeto que facilitam a manutenção e a extensibilidade.

Arquitetura Geral do Sistema

A aplicação segue os princípios da **Inversão de Controle (IoC)** do Spring Framework, utilizando **injeção de dependências** para desacoplar componentes e serviços. A principal funcionalidade está distribuída entre o controlador, que lida com as requisições HTTP, e três serviços distintos: gravação de áudio, conversão de áudio e transcrição. Esta separação de responsabilidades é essencial para garantir a **coesão** e **responsabilidade única** (SRP - Single Responsibility Principle) de cada componente.

1. Controlador HelloController

O controlador é responsável por orquestrar as chamadas aos serviços que realizam a gravação e transcrição do áudio. Através das anotações @PostMapping e @GetMapping, o controlador mapeia as rotas HTTP para os métodos correspondentes.

```
@Controller
public class HelloController {

    @Autowired
    private TranscreverService transcreverService;

    @Autowired
    private GravarAudioFileService gravarAudioFileService;

    @PostMapping("/gravar-audio")
    public String gravarAudio(Model model) {
        try {
            gravarAudioFileService.gravarAudio();
```

```
            String transcricao = transcreverService.transcreverAudio("C:\\Users\\Auro
Neto\\Music\\Teste\\refinado.mp3");
            model.addAttribute("message", transcricao);
            return "Result";
        } catch (Exception e) {
            model.addAttribute("message", "Erro ao processar áudio: " + e.getMessage());
            return "Result";
        }
    }

    @GetMapping("/hello")
    public String hello() {
        return "hello";
    }
}
```

Análise Técnica:

- **Injeção de Dependências via @Autowired**: O Spring gerencia o ciclo de vida dos objetos injetados, garantindo a inversão de controle e promovendo o baixo acoplamento entre os componentes.
- **Tratamento de Exceções**: As exceções são capturadas e uma mensagem amigável é exibida ao usuário, mantendo o feedback adequado em casos de falhas. O uso de try-catch aqui poderia ser refinado para logar exceções utilizando bibliotecas como **SLF4J** para facilitar o monitoramento e análise de erros em produção.

2. Serviço de Conversão de Áudio ConverterWavToMp3Service

A conversão de arquivos WAV para MP3 é realizada pela biblioteca **JAVE (Java Audio/Video Encoder)**. Este serviço encapsula a lógica de conversão, tornando-o reutilizável em diferentes partes da aplicação.

```
public class ConverterWavToMp3Service {

    public void converterToMp3() {
        File wavFile = new File("C:\\Caminho\\Do\\Seu\\Arquivo\\NomeDoArquivo.wav");
        File mp3File = new File("C:\\Caminho\\Do\\Seu\\Arquivo\\NomeDoArquivo.mp3");

        try {
            if (!wavFile.exists()) {
                throw new FileNotFoundException("Arquivo WAV não encontrado.");
            }

            AudioAttributes audioAttributes = new AudioAttributes();
            audioAttributes.setCodec("libmp3lame");
            audioAttributes.setBitRate(64000); // Configuração ajustável conforme necessidade
            audioAttributes.setChannels(1);    // Monofônico para otimização de tamanho de arquivo
            audioAttributes.setSamplingRate(22050); // Taxa de amostragem padrão

            EncodingAttributes encodingAttributes = new EncodingAttributes();
            encodingAttributes.setOutputFormat("mp3");
            encodingAttributes.setAudioAttributes(audioAttributes);
```

```
            Encoder encoder = new Encoder();
            encoder.encode(new MultimediaObject(wavFile), mp3File, encodingAttributes);

        } catch (EncoderException e) {
            throw new RuntimeException("Erro durante a codificação de áudio: " + e.getMessage(),
e);
        }
    }
}
```

Análise Técnica:

- **Biblioteca JAVE**: A JAVE é uma biblioteca madura e eficiente para conversão de arquivos de mídia. A configuração de atributos como bitRate, channels e samplingRate foi otimizada para garantir um bom equilíbrio entre qualidade de áudio e tamanho de arquivo.
- **Tratamento de Exceções Avançado**: Em vez de apenas imprimir a exceção, o código propaga uma exceção personalizada, permitindo que níveis superiores do stack capturem o erro e tomem decisões apropriadas, como registrar logs ou acionar mecanismos de fallback.

3. Serviço de Gravação de Áudio GravarAudioFileService

O serviço de gravação de áudio utiliza a API **Java Sound** para capturar o áudio do microfone e salvar em um arquivo WAV. A gravação é limitada a 10 segundos, porém, essa duração pode ser configurável.

```
@Service
public class GravarAudioFileService {

    private static final int GRAVACAO_DURATION_MS = 10000;

    public void gravarAudio() {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, true);
        File wavFile = new File("C:\\Caminho\\Da\\Pasta\\NomeDoArquivo.wav");

        try (TargetDataLine microphone = AudioSystem.getTargetDataLine(format)) {
            microphone.open(format);
            microphone.start();

            AudioInputStream audioStream = new AudioInputStream(microphone);

            Timer timer = new Timer();
            timer.schedule(new TimerTask() {
                @Override
                public void run() {
                    microphone.stop();
                    microphone.close();
                }
            }, GRAVACAO_DURATION_MS);

            new Thread(() -> {
                try {
```

```
                AudioSystem.write(audioStream, AudioFileFormat.Type.WAVE, wavFile);
            } catch (IOException e) {
                throw new RuntimeException("Erro ao gravar áudio", e);
            }
        }).start();

        Thread.sleep(GRAVACAO_DURATION_MS);
        timer.cancel();
    } catch (LineUnavailableException | InterruptedException e) {
        throw new RuntimeException("Erro durante a captura de áudio: " + e.getMessage(), e);
    }

    new ConverterWavToMp3Service().converterToMp3();
  }
}
```

Análise Técnica:

- **Controle Temporal com Timer**: Utilizando um Timer, o serviço garante que a gravação seja limitada a um período de tempo predefinido. Isso é essencial para evitar gravações excessivamente longas que possam comprometer a eficiência do sistema.
- **Execução em Threads**: A gravação é realizada em uma thread separada, permitindo que o processo principal continue a responder a outros eventos. A utilização de threads é crucial para manter a responsividade da aplicação, especialmente em operações I/O intensivas.

4. Serviço de Transcrição TranscreverService

A transcrição de áudio é realizada por meio da **Google Cloud Speech API**. O serviço lê o arquivo de áudio MP3 e utiliza o modelo de transcrição da Google para converter fala em texto.

```
@Service
public class TranscreverService {

  public String transcreverAudio(String audioFilePath) throws IOException {

    Path path = Paths.get(audioFilePath);
    byte[] audioBytes = Files.readAllBytes(path);
    ByteString audioBytesStr = ByteString.copyFrom(audioBytes);

    try (SpeechClient speechClient = SpeechClient.create()) {

      RecognitionConfig recognitionConfig = RecognitionConfig.newBuilder()
          .setEncoding(RecognitionConfig.AudioEncoding.MP3)
          .setSampleRateHertz(16000)
          .setLanguageCode("pt-BR")
          .build();

      RecognitionAudio recognitionAudio = RecognitionAudio.newBuilder()
          .setContent(audioBytesStr)
          .build();
```

```
            RecognizeResponse response = speechClient.recognize(recognitionConfig,
recognitionAudio);

            return response.getResultsList().stream()
                    .flatMap(result -> result.getAlternativesList().stream())
                    .map(SpeechRecognitionAlternative::getTranscript)
                    .reduce("", (partial, full) -> partial + full + "\n");

        } catch (Exception e) {
            throw new RuntimeException("Erro durante a transcrição de áudio: " + e.getMessage(),
e);
        }
    }
}
```

Análise Técnica:

- **Google Cloud Speech API**: A API é configurada para aceitar arquivos MP3 com taxa de amostragem de 16kHz e realizar transcrições no idioma português brasileiro. A arquitetura permite fácil adaptação a outras linguagens ou formatos de áudio, bastando ajustar os parâmetros da RecognitionConfig.
- **Uso de Streams**: O uso de **Java Streams** torna o código mais conciso e eficiente ao concatenar os resultados da transcrição.

Boas Práticas e Recomendações

1. **Modularidade e Extensibilidade**: A arquitetura orientada a serviços e a separação de responsabilidades permite que cada componente seja facilmente substituível ou estendido. Por exemplo, o serviço de transcrição poderia ser adaptado para usar outro provedor de API sem impactar as demais funcionalidades.
2. **Injeção de Dependências**: O uso de injeção de dependências (IoC) promove baixo acoplamento entre as classes e facilita a testabilidade, permitindo que mocks sejam injetados durante testes unitários.
3. **Gerenciamento de Exceções**: A propagação de exceções personalizadas permite que as falhas sejam tratadas adequadamente em níveis superiores, garantindo que problemas críticos sejam logados e analisados corretamente.
4. **Multithreading e Concorrência**: A gravação de áudio e a manipulação de arquivos de mídia são realizadas em threads separadas para garantir a responsividade da aplicação, evitando o bloqueio da thread principal.

Conclusão

Este projeto apresenta uma solução escalável e eficiente para gravação, conversão e transcrição de áudio. Com a utilização de boas práticas de engenharia de software, como injeção de dependências, modularidade e controle avançado de exceções, a aplicação oferece uma base sólida para futuras expansões ou adaptações para novos cenários.

EducaCiência FastCode para a comunidade