Spotibooks: uma aplicação *web* para organização e recuperação de audiolivros

Fillipe de Oliveira Almeida, Patrícia Nascimento Silva
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
fllp@ufmg.br, patricians@eci.ufmg.br

Resumo

Este artigo apresenta o desenvolvimento da aplicação *web* Spotibooks, concebida para a catalogação e o gerenciamento de audiolivros a partir de um modelo de metadados específico. O projeto surge em resposta ao crescimento do mercado de audiolivros e à consequente necessidade de ferramentas mais robustas e eficientes para organizar, gerenciar e recuperar esse tipo de suporte informacional. O objetivo principal foi implementar, de forma prática, o modelo de metadados proposto na pesquisa de Gonçalves e Nascimento Silva (2024), criando uma solução que otimiza a busca e a organização de acervos de audiolivros. A metodologia adotou princípios da engenharia de *software* moderna, com um planejamento inicial que abrangeu a especificação de requisitos, a modelagem do sistema e o projeto arquitetural, seguindo um processo de desenvolvimento ágil baseado no *framework* Scrum. A arquitetura final do sistema foi implementada como uma Single Page Application (SPA), utilizando Angular no *frontend* e uma API RESTful com Laravel no *backend*, com banco de dados PostgreSQL e armazenamento de arquivos em nuvem. Como resultado, foi entregue uma aplicação *web* funcional que permite o gerenciamento completo de audiolivros e seus metadados, incluindo funcionalidades de busca avançada, filtros dinâmicos e um reprodutor de áudio integrado. O sistema foi validado com a inclusão de audiolivros de domínio público, comprovando a eficácia do modelo em um cenário teórico e prático. Conclui-se que o projeto atingiu seus objetivos, demonstrando, com sucesso, a aplicabilidade do modelo teórico em um produto tecnológico funcional que contribui para a organização, representação e recuperação de audiolivros.

Palavras-chave: Audiolivro, representação, recuperação de informação, engenharia de *software*.

Abstract

This paper presents the development of the Spotibooks web application, designed for cataloging and managing audiobooks based on a specific metadata model. The project arises in response to the growth of the audiobook market and the consequent need for more robust and efficient tools to organize, manage and retrieve this type of informational medium. The main objective was to practically implement the metadata model proposed in the research by Gonçalves and Nascimento Silva (2024), creating a solution that optimizes the search and organization of audiobook collections. The methodology adopted principles of modern software engineering, with initial planning that included requirements specification, system modeling and architectural design, following an agile development process based on the Scrum framework. The final system architecture was implemented as a Single Page Application (SPA) using Angular for the frontend and a RESTful API with Laravel for the backend, with a PostgreSQL database and cloud file storage. As a result, a functional web application that allows for the complete management of audiobooks and their metadata was delivered, including advanced search functionalities, dynamic filters and an integrated audio player. The system was validated with the inclusion of public domain audiobooks, proving the model's effectiveness in a theoretical and practical scenario. It is concluded that the project achieved its objectives, successfully demonstrating the applicability of the theoretical model in a functional technological product that contributes to the organization, representation, and retrieval of audiobooks.

Keywords: Audiobook, representation, information retrieval, software engineering.

1 Introdução

Com a evolução das tecnologias de informação, a evolução da *web* e, mais recentemente, a transformação digital, novos suportes informacionais e ambientes digitais têm sido utilizados para promover o acesso à informação. O audiolivro é um livro em áudio que pode ser ouvido, narrado e interpretado por pessoas voluntárias, profissionais ou pelo próprio autor, contendo sonorizações em suas narrativas (Menezes e Franklin, 2008). Junto ao crescimento do mercado de audiolivros, surge a necessidade de ferramentas mais robustas e eficientes para organizar e gerenciar a informação nesse tipo de suporte. A representação da informação se destaca, na Ciência da Informação, como um campo fundamental do tratamento informacional ao considerar os aspectos de forma e conteúdo dos recursos informacionais, possibilitando uma recuperação eficiente e integrada. Em ambientes digitais, esses processos tendem a ser executados de modo associado, em uma única plataforma, por meio dos metadados (Triques; Arakaki; Castro, 2020). Este estudo teve como objetivo desenvolver um aplicativo *web* focado na catalogação de audiolivros, aplicando os padrões de metadados descritos na pesquisa de Gonçalves e Nascimento Silva (2024). O diferencial deste projeto consistiu na implementação de um produto tecnológico, ou seja, uma aplicação prática que possibilitou o uso desses padrões para otimizar a busca, a organização e o gerenciamento de audiolivros. A abordagem foi centrada em princípios da engenharia de *software*, com foco em modularidade, escalabilidade e boas práticas de desenvolvimento. O projeto foi desenvolvido ao longo das duas disciplinas de monografia (2024/2 e 2025/1), compreendendo as etapas de especificação de requisitos, modelagem, desenvolvimento e validação do sistema, culminando na aplicação funcional descrita neste relatório.

2 Referencial Teórico

A organização e recuperação eficientes de informações em ambientes, plataformas e bibliotecas digitais dependem de metadados bem estruturados. Esses conjuntos de dados descrevem e categorizam informações sobre recursos, facilitando a busca e o gerenciamento. No contexto dos audiolivros, os metadados podem incluir informações como título, autor, narrador, duração e capítulos, elementos importantes para garantir que as grandes coleções de conteúdo em áudio sejam organizadas e facilmente recuperadas. Ao longo dos anos, diversos padrões de metadados foram criados para descrever recursos digitais. Entre os mais reconhecidos, estão:

- **Dublin Core:** desenvolvido em 1995 por uma colaboração internacional na Universidade de Oregon, o Dublin Core consiste em um conjunto de 15 elementos fundamentais que permitem descrever recursos digitais e físicos, facilitando sua busca e recuperação. Este padrão é amplamente utilizado em bibliotecas digitais e repositórios acadêmicos para melhorar a interoperabilidade dos metadados (Weibel, 2005).

- **MARC (Machine-Readable Cataloging):** criado na década de 1960, o MARC foi um dos primeiros padrões para a catalogação de recursos bibliográficos em formato legível por máquina. Desenvolvido pela Biblioteca do Congresso dos Estados Unidos, o MARC tornou-se um padrão amplamente adotado em bibliotecas ao redor do mundo para catalogar livros, periódicos e outros materiais, permitindo a troca de informações catalográficas entre diferentes sistemas (Library of Congress, 2024).

- **ID3:** lançado na década de 1990, o ID3 é um padrão utilizado para armazenar informações sobre arquivos de áudio, como músicas digitais. Ele permite que metadados como título, artista, álbum e gênero sejam incorporados diretamente nos arquivos de áudio, facilitando a organização e a busca dessas mídias em *softwares* de reprodução. O ID3 evoluiu ao longo dos anos, com versões sucessivas, sendo a ID3v2 uma das mais utilizadas atualmente (ID3.org, 2024).

Embora esses padrões sejam amplamente aplicáveis a recursos digitais, podem não atender completamente às necessidades específicas dos audiolivros, que possuem características próprias, como narradores, duração e divisão em capítulos. Dessa forma, a adaptação ou combinação desses padrões, como proposta por Gonçalves e Silva (2024), foi necessária para criar um modelo de metadados eficaz para audiolivros.

Para que o desenvolvimento do sistema de gerenciamento de audiolivros fosse bem-sucedido, foi necessário aplicar princípios sólidos de engenharia de *software*. Segundo Sommerville (2015), o desenvolvimento de *software* moderno envolve a especificação clara de requisitos e o uso de *design* modular para garantir a escalabilidade e a manutenibilidade do sistema. A modularidade permite que diferentes componentes do sistema sejam desenvolvidos de maneira independente e com baixo acoplamento, facilitando futuras expansões e manutenções.

Além disso, práticas como refatoração contínua são essenciais para manter a qualidade do código e garantir que ele se adapte às mudanças ao longo do tempo. Fowler (2004) destaca que a refatoração constante permite a correção de problemas de *design* e melhora a estrutura do *software*, assegurando

que ele continue eficiente, mesmo com o aumento de sua complexidade.

Valente (2020) também reforça a importância de garantir que o *design* de *software* seja baseado em princípios de baixo acoplamento e alta coesão. Isso garante que cada módulo do sistema tenha responsabilidades bem definidas, evitando dependências excessivas entre os componentes. Essas práticas são essenciais para desenvolver um sistema que seja resiliente e flexível o suficiente para evoluir conforme as necessidades do usuário.

O modelo de metadados sugerido por Gonçalves e Silva (2024) foi implementado neste projeto, adaptando padrões como Dublin Core, ID3 e MARC para atender às particularidades dos audiolivros, conforme apresentado na Figura 1. Ao mesmo tempo, o desenvolvimento foi orientado pelos princípios de engenharia de *software* de Sommerville (2015), Fowler (2004) e Valente (2020), com foco em modularidade, refatoração e escalabilidade. Isso garantiu que o sistema de gerenciamento de audiolivros fosse eficiente, flexível e manutenível.

Figura 1: Modelo de metadados para representação de audiolivros
Fonte: Gonçalves e Nascimento Silva (2024).

Para a gestão do projeto, foi adotado o *framework* Scrum, uma escolha alinhada às práticas de desenvolvimento ágil que, segundo Valente (2020), são essenciais para ambientes que exigem flexibilidade e resposta rápida a mudanças. O Scrum permitiu que o projeto fosse dividido em *sprints*, ciclos de desenvolvimento curtos que possibilitaram revisões constantes e entregas incrementais, garantindo que as funcionalidades e os módulos do sistema de audiolivros sejam aprimorados de acordo com um *feedback* contínuo. Essa abordagem ágil contribui para manter o foco nos princípios de modularidade e coesão, fundamentais para um sistema escalável e sustentável a longo prazo.

3 Metodologia

Esta pesquisa, de natureza aplicada, buscou solucionar um problema real, vivenciado em diversas plataformas de *e-books* e audiolivros, a partir de uma proposta tecnológica, com foco nos metadados utilizados para a representação e posterior recuperação dos audiolivros. Assim, os procedimentos metodológicos abrangeram as etapas de planejamento, desenvolvimento e validação, seguindo princípios da engenharia de *software* moderna e práticas ágeis. O processo iniciou-se com a especificação de requisitos, seguida da modelagem, da definição do projeto arquitetural, do desenvolvimento e, por fim, da etapa de testes e validação.

A definição dos requisitos é uma etapa inicial e essencial no desenvolvimento do sistema, pois orienta a construção e garante que a aplicação atenda às expectativas dos usuários. Os requisitos são organizados em duas categorias principais: funcionais, que descrevem as funcionalidades específicas do sistema; e não funcionais, que estabelecem critérios de qualidade, desempenho e restrições técnicas.

3.1 Requisitos funcionais

Os requisitos funcionais especificam as principais operações que o sistema deve realizar:

1. **Gerenciamento de audiolivros**
   - O sistema deve permitir o CRUD (cadastro, visualização, edição e exclusão) de audiolivros com os seguintes campos de metadados:
     - Informações Essenciais: número de identificação (gerado automaticamente), título, subtítulo, autor/criador, narrador, edição, local, editora, selo da editora, data, série e volume.
     - Informações Complementares: idioma, idioma original, tradutor, direitos autorais, ISBN ou identificador semelhante e mercados disponíveis.
     - Informações de Conteúdo e Classificação: resumo, assunto, classificação indicativa, público-alvo, gênero/categoria/tipo e capa.
     - Informações Técnicas (vinculadas ao audiolivro como um todo): formato, tamanho e duração total.
2. **Gerenciamento de capítulos**
   - Deve ser possível o CRUD (cadastro, visualização, edição e exclusão) de capítulos. Os capítulos são vinculados a um audiolivro e incluem os seguintes metada-

dos: identificador do audiolivro, número da faixa, título e duração.

3. **Gerenciamento de arquivos de áudio**
   - O sistema deve permitir o CRUD (cadastro, visualização, edição e exclusão) de arquivos de áudio que estão relacionados a um capítulo de um audiolivro.
   - O sistema deve calcular automaticamente a duração total do audiolivro com base na soma das durações dos capítulos.
4. **Validação de dados**
   - O sistema deve validar os seguintes aspectos durante o cadastro:
     - Preenchimento de todos os campos obrigatórios.
     - Consistência dos capítulos (duração coerente dos arquivos de áudio, por exemplo).
     - Formato dos identificadores (ISBN ou código BISAC).
5. **Busca rápida**
   - O sistema deve permitir que os usuários, por meio de uma barra de pesquisa, busquem audiolivros por palavras-chave. Serão listados apenas audiolivros que possuem essas palavras em algum dos seus metadados.
6. **Busca avançada**
   - O sistema deve permitir que os usuários façam uma busca avançada entre os audiolivros, podendo selecionar quais são os valores buscados para cada campo em específico.
     - Serão definidos os campos com maior prevalência para busca dentro de cada grupo de metadados e cada um destes terá um filtro correspondente.
7. **Reprodução e *download* de audiolivro**
   - O sistema deve permitir a reprodução ou o *download* de audiolivros diretamente na aplicação, oferecendo um *player* de áudio integrado com as seguintes funcionalidades:
     - Reprodução de capítulos ou de arquivo completo.
     - Controles básicos, como reproduzir/pausar, avançar/retroceder, ajustar volume e exibir barra de progresso.
   - Será usado um *player* de áudio embutido na página, utilizando bibliotecas para aplicações *web*. O *player* será responsivo, adaptando-se aos diferentes dispositivos nos quais a aplicação poderá ser utilizada.

3.2 Requisitos não funcionais

Os requisitos não funcionais tratam das características do sistema que garantem sua qualidade, confiabilidade e usabilidade:

1. **Desempenho**
   - O sistema deve suportar até 1.000 usuários simultâneos sem degradação perceptível de desempenho.
   - As consultas realizadas na funcionalidade de busca devem ser processadas em, no máximo, 2 segundos para garantir uma experiência ágil ao usuário.
2. **Usabilidade**
   - A interface deve ser responsiva, adaptando-se a diferentes dispositivos, como *desktops*, *tablets* e *smartphones*.
   - O *design* deve seguir os princípios de usabilidade, proporcionando uma experiência intuitiva, especialmente para usuários não técnicos.
3. **Escalabilidade**
   - A arquitetura do sistema deve permitir a adição de novos módulos e funcionalidades com impacto mínimo no código existente.
4. **Segurança**
   - O sistema deve implementar autenticação e autorização robustas para proteger os dados dos usuários.
5. **Confiabilidade e disponibilidade**
   - O sistema deve garantir 99,9% de disponibilidade, com tolerância a falhas e suporte a *backups* automáticos.
6. **Portabilidade**
   - O sistema deve ser compatível com os principais navegadores modernos.
7. **Manutenibilidade**
   - O código-fonte deve ser modular e bem documentado para facilitar futuras manutenções e extensões do sistema.
8. **Infraestrutura para desenvolvimento e hospedagem**

- O sistema será desenvolvido em um ambiente flexível, com suporte para escalabilidade, o que permitirá ajustes e crescimento conforme as necessidades do projeto.
- O ambiente de produção será baseado em infraestrutura de nuvem, garantindo alta disponibilidade e facilidade de manutenção.
- A solução será hospedada em servidores adequados para suportar a carga de usuários esperada e será projetada para garantir a continuidade do serviço em caso de falhas.

As seções a seguir, 4 e 5, apresentam aspectos da modelagem e do projeto arquitetural, ambos essenciais para o planejamento do projeto de implementação.

4 Modelagem de *software*

A modelagem de *software* é fundamental no desenvolvimento do sistema, pois permite visualizar e compreender, de forma clara, a estrutura e o funcionamento do sistema antes de sua implementação. Alguns diagramas foram escolhidos para essa representação e são apresentados nas seções 4.1 a 4.3.

4.1 Diagrama de casos de uso

Os diagramas de casos de uso visam modelar as interações entre os usuários e o sistema, ou seja, as funcionalidades que o sistema deve oferecer. Cada caso de uso representa uma ação que o usuário pode realizar, como "cadastrar um audiolivro", "realizar uma busca"ou "reproduzir o audiolivro"(Figura 2).

4.2 Diagrama de classes

O diagrama de classes reflete a estrutura estática do sistema, detalhando as entidades principais, seus atributos e os relacionamentos entre elas. O sistema foi projetado com o intuito de organizar e gerenciar informações sobre audiolivros de forma eficiente e, para isso, foi necessário criar diversas classes, cada uma representando diferentes aspectos dos metadados associados a um audiolivro.

A classe principal Audiolivro contém os dados essenciais do audiolivro, divididos em quatro categorias de metadados: Informações Essenciais, Informações Complementares, Informações de Conteúdo e Informações Técnicas. Além disso, a classe Audiolivro está relacionada com as classes Capítulo e ArquivoAudio, que representam, respectivamente, as partes do audiolivro e os arquivos de áudio correspondentes (Figura 3).

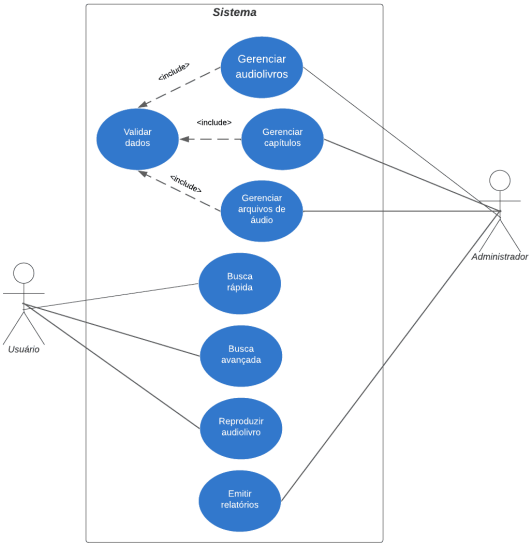

Figura 2: Diagrama de caso de uso
Fonte: elaborado pelo autor (2025).

4.3 Diagrama de sequência

O diagrama de sequência mostra a interação entre os objetos do sistema ao longo do tempo, em resposta a um evento específico. Ele detalha a ordem em que as mensagens são trocadas entre os objetos para realizar uma determinada tarefa. Para fins de exemplificação, foram selecionadas duas ações: o cadastro de audiolivros (Figura 4) e a reprodução de audiolivros (Figura 5). Essas ações destacam os fluxos mais relevantes, evidenciando as etapas necessárias para realizar cada operação com sucesso.

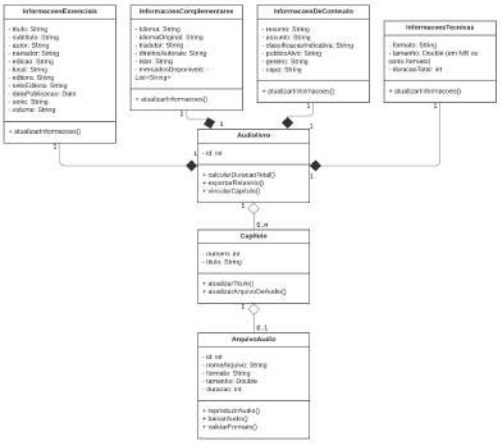

Figura 3: Diagrama de classes
Fonte: elaborado pelo autor (2025).

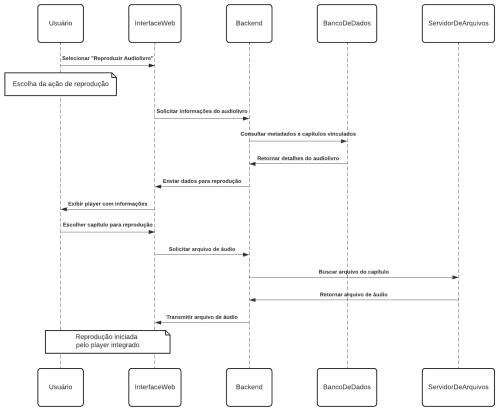

Figura 5: Diagrama de sequência para a funcionalidade "Reproduzir audiolivro"
Fonte: elaborado pelo autor (2025).

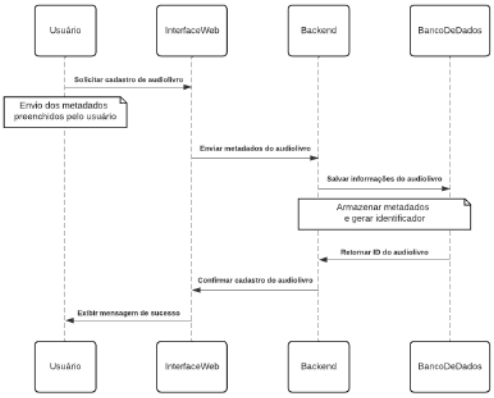

Figura 4: Diagrama de sequência para a funcionalidade "Cadastrar audiolivro"
Fonte: elaborado pelo autor (2025).

5 Projeto arquitetural

O projeto arquitetural do sistema adota o padrão Model-View-Controller (MVC), que é amplamente utilizado em sistemas *web* por facilitar a separação de responsabilidades entre as camadas de apresentação, controle e lógica de negócios. Essa abordagem é fundamentada nas definições apresentadas por Sommerville (2015), que destaca o MVC como um padrão essencial para a modularidade e escalabilidade em sistemas complexos. De acordo com Sommerville (2015), o modelo permite isolar as responsabilidades principais, garantindo que mudanças em uma camada tenham impacto mínimo nas demais, o que é particularmente útil em sistemas com requisitos dinâmicos e interfaces interativas.

Neste projeto, a camada de Modelo é responsável por gerenciar os dados e a lógica de negócio. Essa camada foi implementada no *backend* com o *framework* Laravel[1], que oferece suporte nativo à manipulação de modelos, à abstração da interação com o banco de dados relacional e ao gerenciamento de regras de negócio, como permissões de acesso e rastreamento de metadados. A camada de Visão é representada pela interface do usuário, desenvolvida com o *framework* Angular[2], que permitiu a criação de Single Page Applications (SPAs) dinâmicas e responsivas. Por fim, a camada de Controle, central no padrão MVC, foi implementada no Laravel, atuando como intermediária entre as camadas de Modelo e Visão, processando requisições vindas do *frontend*, aplicando a lógica de negócio e retornando os dados esperados por meio de Application Programming Interface (API) Representational State Transfer (RESTful).

Pressman (2020) reforça os benefícios do MVC ao apontar que o padrão promove a reutilização de componentes e simplifica o processo de desenvolvimento ao separar as preocupações de apresentação e lógica. Essa característica é particularmente relevante para o projeto em questão, que prevê a expansão futura do sistema e integrações adicionais. A adoção do MVC neste projeto, portanto, não apenas atende às necessidades imediatas, mas também prepara o sistema para uma evolução contínua. A Figura 6 apresenta a arquitetura do sistema, as interações e as tecnologias utilizadas.

[1]LARAVEL. The PHP Framework for Web Artisans. Disponível em: https://laravel.com. Acesso em: 29 jun. 2025.

[2]ANGULAR. The web development framework for building the future. Disponível em: https://angular.dev. Acesso em: 29 jun. 2025.

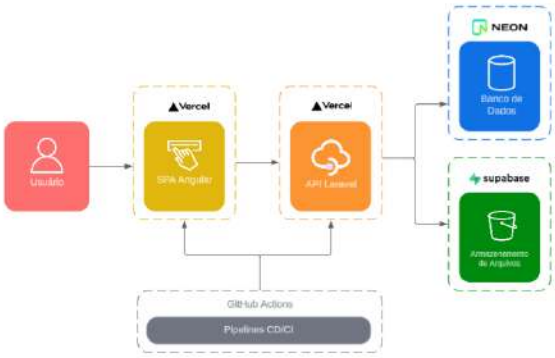

Figura 6: Projeto arquitetural do sistema
Fonte: elaborado pelo autor (2025).

5.1 *Backend*

O *backend* foi desenvolvido com Laravel, um *framework* PHP reconhecido por sua simplicidade na criação de APIs RESTful, sendo responsável por implementar toda a lógica de negócio e as operações de CRUD.

A escolha para a implantação foi a plataforma Vercel[3], que opera sobre uma arquitetura *serverless*, para abstrair do desenvolvedor toda a complexidade de provisionar, configurar um servidor *web*, gerenciar o sistema operacional, aplicar patches de segurança e planejar a escalabilidade.

Ao usar a Vercel, o desenvolvedor se concentra apenas no código da aplicação Laravel. A plataforma se encarrega de executar esse código em um ambiente otimizado, que escala automaticamente conforme a demanda e possui um custo atrelado diretamente ao uso. Essa abordagem foi decisiva para o projeto, pois permitiu que o foco permanecesse na construção do produto tecnológico.

5.2 *Frontend*

O *frontend* foi desenvolvido como uma Single Page Application (SPA) com Angular 19, um *framework* ideal para a criação de interfaces dinâmicas, interativas e responsivas.

Suas responsabilidades incluem:

- Renderizar todas as interfaces de usuário para gerenciar audiolivros e interagir com o sistema.
- Realizar o consumo e a manipulação dos dados em tempo real, por meio de chamadas HTTP para a API do *backend*.
- Garantir um *design* responsivo, oferecendo uma boa experiência de uso em diferentes dispositivos, como *desktops* e celulares.

[3]VERCEL. Develop. Preview. Ship.. Disponível em: https://vercel.com. Acesso em: 29 jun. 2025.

A hospedagem do *frontend* foi realizada também na Vercel, que oferece integração nativa com repositórios Git (GitHub) e um *pipeline* de Continuous Integration/Continuous Delivery (CI/CD), otimizando a entrega de conteúdo globalmente a partir de sua Content Delivery Network (CDN).

5.3 Banco de dados

Para a persistência dos dados, o sistema utiliza o PostgreSQL, um sistema de gerenciamento de banco de dados relacional de código aberto, hospedado na plataforma Neon DB[4] em um modelo *serverless*. A escolha se deu pela robustez do PostgreSQL para gerenciar dados estruturados e pelas vantagens operacionais e de custo que a Neon oferece.

É fundamental destacar a função específica do banco de dados nesta arquitetura: ele foi projetado para armazenar exclusivamente as tabelas com os metadados do sistema. Isso inclui informações como título, autor, narrador, duração e outros campos textuais e numéricos que descrevem os audiolivros.

Os arquivos de áudio (.mp3, por exemplo), por sua vez, não são armazenados nesse banco de dados. Em vez disso, as tabelas contêm apenas uma coluna de texto que guarda a referência para a localização de cada arquivo de áudio no serviço de armazenamento de objetos (detalhado na próxima seção).

5.4 Armazenamento de arquivos

Para o armazenamento dos arquivos de áudio, a solução escolhida foi o Supabase Storage[5]. A decisão foi motivada pela simplicidade e pela rica gama de funcionalidades que a plataforma oferece ao desenvolvedor, facilitando as operações de *upload*, a definição de políticas de acesso e a geração de URLs seguras para a reprodução dos áudios. Embora a interação do *backend* do sistema seja exclusivamente com o ecossistema do Supabase, a plataforma utiliza a infraestrutura robusta do Amazon S3[6] para armazenar fisicamente os arquivos.

Dessa forma, o projeto herda a escalabilidade e robustez da infraestrutura do S3, porém a interação ocorre por meio da API do Supabase, que é mais simples e focada na experiência do desenvolvedor. Além disso, o uso do plano gratuito do Supabase forneceu um ambiente controlado e sem custos, eliminando a possibilidade de cobranças inesperadas

[4]NEON. The serverless PostgreSQL. Disponível em: https://neon.tech. Acesso em: 29 jun. 2025.

[5]SUPABASE. Build in a weekend. Scale to millions. Disponível em: https://supabase.com. Acesso em: 29 jun. 2025.

[6]AMAZON WEB SERVICES. Cloud Object Storage - Amazon S3. Disponível em: https://aws.amazon.com/s3/. Acesso em: 29 jun. 2025.

que poderiam surgir com o uso direto de serviços de nuvem, tornando a solução ideal para o cronograma e os recursos do projeto.

5.5 Comunicação do sistema

A automação da implantação foi configurada a partir do GitHub Actions. O *pipeline* de CI/CD automatiza o *deploy* de qualquer alteração no código: o *frontend* é publicado no ambiente da Vercel, e o *backend* é atualizado como uma função *serverless* na mesma plataforma. A comunicação do sistema flui de maneira integrada: o *frontend* em Angular realiza chamadas para a API RESTful em Laravel, que, por sua vez, processa as requisições, consulta o banco de dados na NeonDB e gerencia os arquivos no S3 via Supabase.

6 Resultados

A transição do planejamento para o desenvolvimento marcou a fase de materialização do projeto. O projeto seguiu uma abordagem de desenvolvimento iterativa, na qual as funcionalidades foram construídas e refinadas em ciclos, permitindo a integração contínua de componentes e a validação constante do produto. Esta seção detalha a implementação das funcionalidades centrais do sistema, que constituem o resultado prático do projeto.

6.1 Gerenciamento do acervo (CRUD)

O núcleo do sistema é a sua capacidade de gerenciar o ciclo de vida completo (organização, representação e recuperação) de um audiolivro. Foi implementada a funcionalidade de CRUD (Cadastro, Leitura, Atualização e Exclusão) tanto para os audiolivros quanto para seus capítulos associados.

- Interface de cadastro e edição: no *frontend*, foram criados formulários reativos em Angular para a inserção e edição de dados. Esses formulários são estruturados para capturar todas as categorias de metadados do modelo proposto: Informações Essenciais, Complementares, de Conteúdo e Técnicas. A interface guia o usuário no preenchimento dos campos, aplicando validações para garantir a consistência dos dados, como o preenchimento de campos obrigatórios.
- Integração com a API: ao submeter um formulário, a aplicação Angular envia uma requisição HTTP para a API em Laravel. O *backend* valida os dados recebidos, processa as informações e realiza a persistência no banco de dados PostgreSQL. A mesma lógica se aplica à atualização e exclusão de registros.

6.2 Recuperação da informação (busca rápida e avançada)

Para que o acervo seja útil, a recuperação da informação precisa ser eficiente. O sistema implementa duas modalidades de busca:

- Busca rápida: disponível na tela principal, esta funcionalidade consiste em um campo de texto único. Quando o usuário insere uma palavra-chave, a API realiza uma consulta simples, buscando o termo nos campos título e autor.
- Busca avançada: esta é uma ferramenta mais poderosa, que permite ao usuário construir consultas estruturadas. A interface possibilita a criação de múltiplos filtros, na qual, para cada filtro, o usuário pode selecionar um campo específico de metadado (ex: "Narrador"), um operador de comparação (ex: "contém") e o valor a ser buscado.

6.3 Interface de reprodução de áudio

A funcionalidade de reprodução de audiolivros foi implementada para ser integrada e funcional.

- *Player* de áudio: a interface utiliza um *player* de áudio embutido, construído com tecnologias *web* padrão (HTML5 Audio) e controlado por componentes Angular.
- Segurança e desempenho: ao solicitar a reprodução, o *frontend* não acessa o arquivo de áudio diretamente. Ele requisita à API uma URL de acesso segura e temporária, que o *backend* gera por meio do Supabase. Isso protege os arquivos e otimiza a entrega.
- Controles e navegação: o *player* oferece os controles básicos de reprodução (reproduzir/pausar, avançar/retroceder, volume) e exibe uma barra de progresso. Adicionalmente, uma lista de capítulos é exibida, permitindo que o usuário navegue diretamente para partes específicas do audiolivro, melhorando significativamente a experiência de uso.

6.4 Design responsivo e usabilidade

Conforme o requisito de usabilidade, a interface da aplicação foi projetada para ser totalmente responsiva. Utilizando técnicas modernas de Cascading Style Sheet (CSS) e a arquitetura de componentes do Angular, o layout se adapta fluidamente a diferentes dispositivos, incluindo *desktops*, *tablets* e *smartphones*, garantindo uma experiência de usuário consistente, mesmo para usuários não técnicos.

6.5 Testes e validação

Para garantir a qualidade, a confiabilidade e o correto funcionamento da aplicação, foi conduzida uma abordagem de testes estruturada, seguida por uma validação prática do sistema com dados reais. Esta etapa foi fundamental para assegurar que o *software* atendesse aos requisitos definidos e identificar se o modelo de metadados adotado é eficaz em um cenário de uso concreto.

6.5.1 Estratégia de testes

A estratégia de testes focou em verificar os componentes do sistema em diferentes níveis. Em um primeiro nível, foram realizados testes unitários para validar as menores partes isoladas da aplicação. No *backend* em Laravel, isso envolveu testar as regras de negócio nos *models* e *services*, enquanto no *frontend* em Angular, os componentes foram testados para verificar sua renderização e comportamento. Em um segundo nível, o foco foi nos testes de integração, que garantiram a correta comunicação entre o *frontend* e a API *backend*. Foram validados os principais fluxos de dados, como o envio de um formulário de cadastro de audiolivro, verificando se a requisição HTTP era enviada corretamente pelo Angular e se a API em Laravel processava os dados e retornava a resposta esperada.

6.5.2 Validação funcional e do modelo de metadados

A validação final do sistema foi realizada com o objetivo de testar as funcionalidades em um cenário prático e, principalmente, comprovar a eficácia do modelo de metadados implementado. Para isso, o sistema foi populado com conteúdo real de audiolivros de domínio público. A fonte para este conteúdo foi o site LibriVox, um conhecido repositório de audiolivros gratuitos. Foram utilizados alguns capítulos de obras clássicas da literatura brasileira, como "Dom Casmurro"e "Memórias Póstumas de Brás Cubas", de Machado de Assis.

O processo consistiu em utilizar o painel administrativo do Spotibooks para cadastrar essas obras, preenchendo os campos de metadados conforme o modelo, como título, autor, narrador, idioma, gênero, entre outros. Em seguida, as funcionalidades da interface pública, como a busca rápida e a busca avançada, foram utilizadas para localizar e filtrar esses registros. A capacidade de catalogar com sucesso esses exemplos reais e, posteriormente, recuperá-los de forma precisa por meio dos filtros, confirmou tanto a corretude funcional da aplicação quanto a adequação e a utilidade prática do modelo de metadados para a organização e representação de audiolivros.

7 Spotibooks

Esta seção apresenta os resultados práticos do projeto: a aplicação *web* Spotibooks, que está funcional e pode ser acessada publicamente a partir do endereço https://spotibooks.vercel.app/. O objetivo é demonstrar como a arquitetura e as tecnologias descritas foram utilizadas para construir uma solução funcional que atende aos requisitos levantados na fase de planejamento. A aplicação é dividida em duas grandes áreas: a interface pública, para os usuários, e o painel administrativo, para o gerenciamento do conteúdo.

7.1 Interface pública: descoberta e consumo

A interface destinada ao usuário final foi projetada para a descoberta e o consumo dos audiolivros. Tela principal e busca rápida: A página inicial da aplicação exibe o catálogo de audiolivros e uma barra de "busca rápida", que permite ao usuário pesquisar por termos presentes no título ou autor da obra (Figura 7).

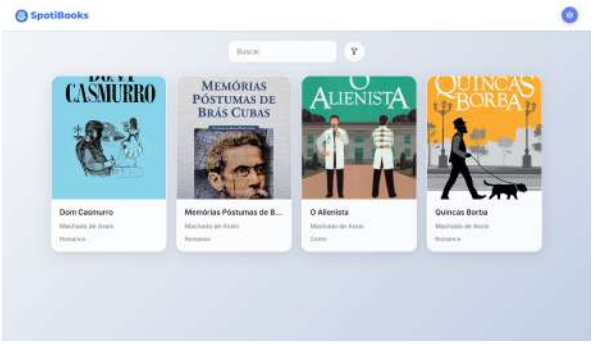

Figura 7: Tela principal da aplicação Spotibooks
Fonte: dados da pesquisa (2025).

Detalhes do audiolivro: ao selecionar uma obra, uma tela exibe todos os seus metadados de forma estruturada, seguindo as quatro categorias do modelo proposto (Essenciais, Complementares, de Conteúdo e Técnicas), bem como sua lista de capítulos (Figura 8).

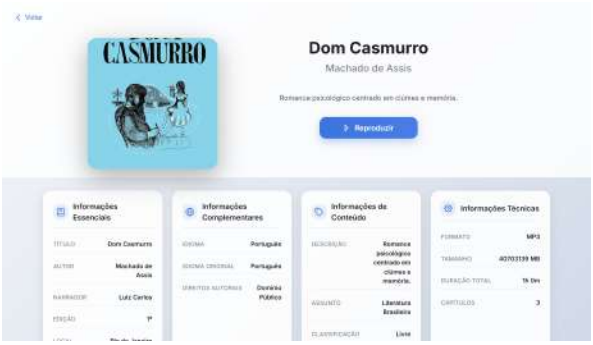

Figura 8: Tela de detalhes do audiolivro
Fonte: dados da pesquisa (2025).

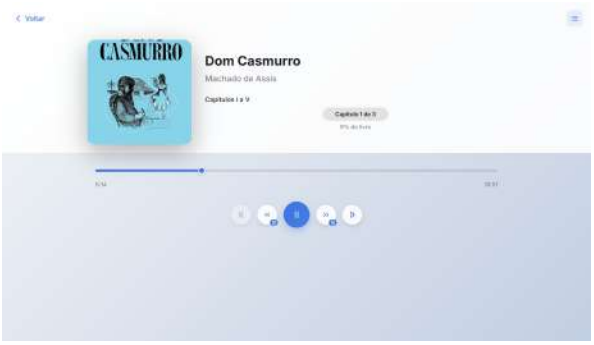

Figura 10: Tela de reprodução do audiolivro
Fonte: dados da pesquisa (2025).

Busca avançada: o sistema oferece uma ferramenta de "busca avançada", pela qual é possível combinar múltiplos campos de metadados para refinar os resultados e encontrar obras com alta precisão (Figura 9).

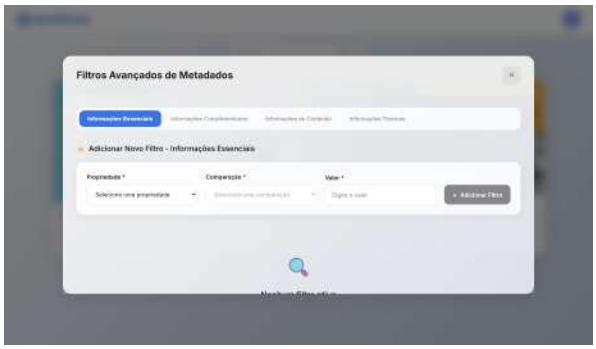

Figura 9: Interface da funcionalidade de busca avançada
Fonte: dados da pesquisa (2025).

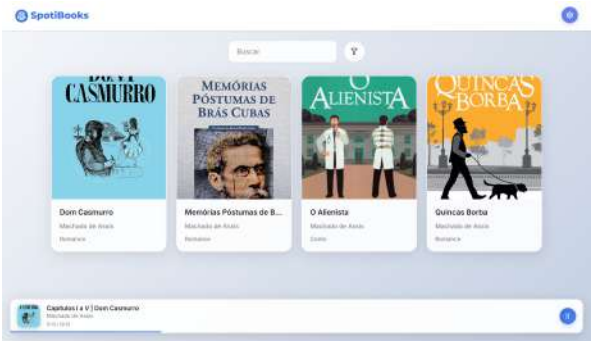

Figura 11: Tela principal da aplicação com miniplayer
Fonte: dados da pesquisa (2025).

Player de áudio integrado: a experiência de consumo do conteúdo foi projetada para ser contínua e flexível. Ao iniciar a reprodução, o usuário tem acesso a um *player* em tela cheia com todos os controles principais, porém este *player* pode ser fechado sem interromper o áudio, transformando-se em um *miniplayer* permanente, que fica fixo na interface. Isso permite que o usuário continue navegando por outras partes da aplicação enquanto o áudio continua rodando em segundo plano (Figuras 10 e 11).

7.2 Painel administrativo: gerenciamento de conteúdo

Para a gestão do acervo, foi desenvolvida uma área administrativa protegida, na qual o administrador pode realizar as operações de CRUD (Cadastro, Leitura, Atualização e Exclusão) sobre os audiolivros e seus capítulos, atendendo a um requisito funcional central do projeto.

Visualização e gestão do acervo: a tela principal do painel administrativo exibe uma lista de todos os audiolivros catalogados, oferecendo ao administrador uma visão geral do acervo e acesso rápido às opções de editar e excluir cada registro (Figura 12).

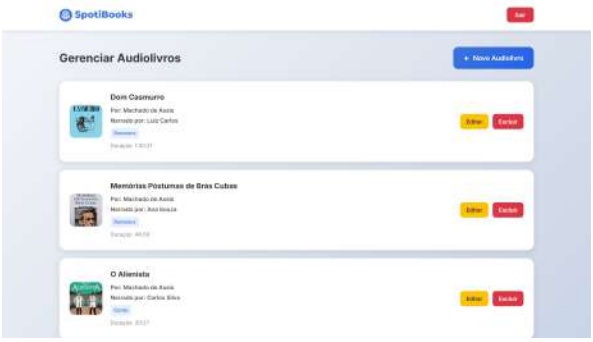

Figura 12: Tela principal do painel administrativo
Fonte: dados da pesquisa (2025).

Cadastro e edição de audiolivros: o painel contém formulários para o cadastro de novas obras. Esses formulários foram projetados para capturar todos os campos de metadados definidos no modelo, permitindo um registro completo e detalhado. A mesma interface é utilizada para a edição de audiolivros existentes (Figura 13).

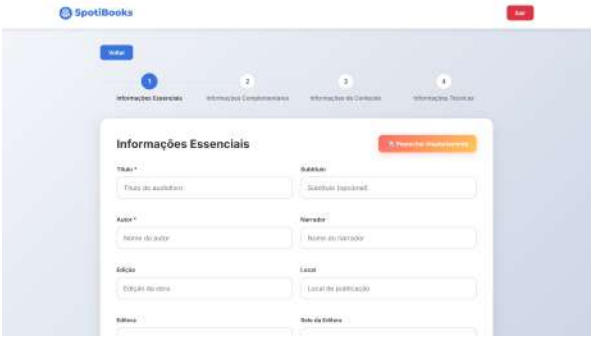

Figura 13: Formulário de cadastro e edição de audiolivro
Fonte: dados da pesquisa (2025).

Gerenciamento de capítulos e arquivos de áudio: após cadastrar um audiolivro, o administrador pode associar seus respectivos capítulos, incluindo o *upload* dos arquivos de áudio para cada um deles. O sistema permite a edição e a exclusão tanto dos capítulos quanto dos arquivos (Figura 14).

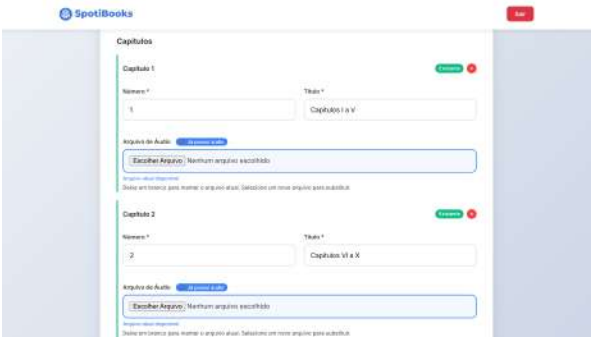

Figura 14: Gerenciador de capítulos do audiolivro
Fonte: dados da pesquisa (2025).

8 Considerações finais

Este relatório apresentou o ciclo completo de desenvolvimento de um sistema *web*, o Spotibooks, produto tecnológico projetado para aplicar um modelo de metadados específico para a organização, representação e recuperação de audiolivros. O principal objetivo do projeto foi criar uma solução prática que utilizasse o modelo definido na pesquisa de Gonçalves e Nascimento Silva (2024), a fim de facilitar a busca, o gerenciamento e o acesso a informações relevantes sobre audiolivros. Este objetivo foi alcançado com a entrega de uma aplicação funcional, que valida a eficácia do modelo teórico em um cenário de uso real, unindo teoria e prática.

O planejamento foi efetuado na disciplina Monografia I, cursada em 2024/2, e a implementação, realizada na disciplina Monografia II, cursada em 2025/1, ambos orientados pela professora Patrícia Nascimento Silva.

O planejamento, que seguiu os princípios da engenharia de *software*, estabeleceu as bases para as etapas de desenvolvimento, testes e validação do sistema. O modelo de metadados serviu como um alicerce para a definição das funcionalidades-chave, como a catalogação detalhada, a busca por metadados e os filtros avançados. A aplicação final demonstra, de forma concreta, a aplicabilidade do modelo para a representação e recuperação de conteúdo, gerando um produto tecnológico que poderá ser expandido e refinado.

Foram implementadas as funcionalidades essenciais para o gerenciamento do acervo, incluindo um painel administrativo para as operações de CRUD (Cadastro, Leitura, Atualização e Exclusão) de audiolivros e seus respectivos capítulos. Para a recuperação da informação, o sistema conta com uma busca rápida por palavra-chave e uma ferramenta de busca avançada que permite a filtragem por múltiplos campos de metadados. A experiência de consumo do conteúdo é viabilizada por uma interface pública com um reprodutor de áudio integrado, que oferece controles básicos de reprodução, navegação por capítulos e um *miniplayer* permanente que permite ao usuário navegar pelo *site* sem interromper o áudio. Todo o sistema foi desenvolvido com um *design* responsivo, garantindo a usabilidade em diferentes dispositivos.

Reconhece-se que a versão atual (1.0), embora funcional, representa uma primeira versão simplificada do sistema. Ela implementa uma base robusta e um excelente ponto de partida para futuras evoluções que podem enriquecer significativamente a experiência do usuário com os audiolivros. A principal linha de melhoria seria a implementação de um sistema de *login* de usuário. Isso abriria caminho para uma série de novas funcionalidades, como a criação de perfis personalizados em que cada usuá-

rio poderia manter um histórico de audiolivros ouvidos e listas de favoritos. Com base nesses dados, o sistema poderia oferecer recomendações personalizadas, sugerindo novas obras alinhadas aos interesses de cada um. Adicionalmente, funcionalidades sociais, como a possibilidade de avaliar e escrever resenhas, poderiam ser implementadas para criar uma comunidade em torno do acervo. Além das melhorias focadas no usuário, outras evoluções poderiam incluir o desenvolvimento de aplicativos móveis nativos e a integração com APIs de editoras para a importação automática de metadados.

Conclui-se, portanto, que o projeto atingiu seus objetivos com sucesso, ilustrando a sinergia entre a Ciência da Informação e a Engenharia de *Software* na criação de ferramentas que melhoram a organização e o acesso ao conhecimento na era digital.

Referências

[1] FOWLER, M. *Refatoração: aperfeiçoando o projeto de código existente*. Porto Alegre: Bookman, 2004.

[2] GONÇALVES, S. S.; NASCIMENTO SILVA, P. Representação de audiolivros: aportes nos padrões de metadados. *Encontros Bibli*, v. 29, 2024: e98860. DOI: `https://doi.org/10.5007/1518-2924.2024.98860`.

[3] ID3.ORG. ID3v2 specifications. Disponível em: `http://id3.org`. Acesso em: 02 nov. 2024.

[4] LIBRARY OF CONGRESS. MARC 21 overview. Disponível em: `https://www.loc.gov/marc/overview/`. Acesso em: 02 nov. 2024.

[5] MENEZES, N. C.; FRANKLIN, S. Audiolivro: uma importante contribuição tecnológica para os deficientes visuais. *Ponto de Acesso*, Salvador, v. 2, n. 3, p. 58-72, dez. 2008. Disponível em: `https://brapci.inf.br/index.php/res/download/98646`. Acesso em: 26 out. 2024.

[6] PRESSMAN, Roger S. *Engenharia de Software: uma abordagem profissional*. 8. ed. Porto Alegre: AMGH, 2020.

[7] SOMMERVILLE, I. *Software engineering*. 10th ed. Boston: Pearson, 2015.

[8] TRIQUES, M. L.; ARAKAKI, A. C. S.; CASTRO, F. F. de. Aspectos da representação da informação na curadoria digital. *Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação*, Santa Catarina, v. 25, p. 01-21, 2020. Disponível em: `https://periodicos.ufsc.br/index.php/eb/article/view/1518-2924.2020.e69898`. Acesso em: 2 ago. 2023.

[9] VALENTE, M. T. *Engenharia de software moderna: princípios e práticas para desenvolvimento de software com produtividade*, Editora: Independente, 2020.

[10] WEIBEL, S. L. Dublin Core Metadata Initiative. In: *Metadata*. 2005.