Diário Cultural:
Sistema de Acompanhamento de Leitura e Audiovisual

Enzo Cauã da S. Barbosa

[1]Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – BA, Brasil – 44036-900

caua7uefs@gmail.com

Abstract. *This report presents the first phase of the development of the Diário Cultural application, aimed at registering and evaluating books, movies, and series consumed by users. In this initial stage, the backend was implemented using the Java programming language, according to the project requirements. The object-oriented modeling included the definition of classes, use cases, unit and integration tests. Although no specific design pattern has been adopted yet, good development practices such as package separation were applied. The implemented features were successfully validated, and limitations such as lack of data persistence and full exception handling will be addressed in the next phases.*

Resumo. *Este relatório apresenta a primeira fase do desenvolvimento da aplicação Diário Cultural, voltada ao registro e à avaliação de livros, filmes e séries consumidos pelo usuário. Nesta etapa inicial, foi implementado o back-end da aplicação utilizando a linguagem Java, conforme os requisitos do projeto. A modelagem orientada a objetos incluiu a definição de classes, casos de uso, testes de unidade e integração. Embora ainda não tenha sido adotado um padrão de projeto formal, foram aplicadas boas práticas como a divisão em pacotes. As funcionalidades desenvolvidas foram validadas com sucesso, e limitações como ausência de persistência de dados e tratamento completo de exceções serão abordadas nas próximas fases.*

1. Introdução

O diário cultural é uma ferramenta utilizada por leitores e espectadores para registrar suas experiências com obras artísticas, como livros, filmes e séries. Por meio de anotações, avaliações e reflexões, o diário permite acompanhar o próprio desenvolvimento intelectual, relembrar conteúdos marcantes e estimular uma relação mais consciente com a produção cultural consumida.

Com o crescimento das plataformas digitais e a facilidade de acesso a conteúdos diversos, o consumo cultural tornou-se cada vez mais intenso e acelerado [Aderaldo et al. 2020]. No entanto, essa abundância pode fazer com que as experiências se tornem superficiais ou rapidamente esquecidas. Diante desse cenário, torna-se relevante o uso de recursos que incentivem a organização e a valorização dessas vivências.

A aplicação *Diário Cultural* foi desenvolvida com esse propósito: oferecer aos usuários um espaço digital para registrar o que leram e assistiram, avaliar obras e anotar

suas impressões. Assim, promove-se a construção de um acervo pessoal de experiências culturais, incentivando a reflexão e o crescimento individual.

Este relatório apresenta a primeira fase do desenvolvimento da aplicação *Diário Cultural*, com ênfase na modelagem e implementação do backend. Esta etapa contempla a definição dos requisitos do sistema, a elaboração dos diagramas de casos de uso e de classes, bem como a codificação das principais funcionalidades da lógica de negócio. Foram realizados testes de unidade e integração para validar o comportamento dos módulos desenvolvidos.

O desenvolvimento foi conduzido utilizando a linguagem de programação Java, conforme especificado no enunciado do projeto, priorizando boas práticas de orientação a objetos, modularidade e testabilidade do código. As próximas seções deste relatório apresentam a fundamentação teórica, a metodologia adotada, os resultados obtidos e as conclusões relativas a esta etapa inicial do projeto.

2. Fundamentação Teórica

O desenvolvimento da aplicação Diário Cultural baseia-se em princípios da programação orientada a objetos (POO), que promove a organização do código em classes e objetos com responsabilidades bem definidas, facilitando a modularidade, a reutilização e a manutenção do software [Henrique 2023].

Além disso, o projeto adota práticas de engenharia de software, como a modelagem por meio de diagramas de casos de uso e diagramas de classes, que auxiliam na compreensão dos requisitos do sistema e na estruturação das funcionalidades.

Para garantir a confiabilidade do sistema, foram utilizados testes de unidade e integração, técnicas fundamentais para validar o comportamento correto de componentes isolados e do sistema como um todo.

A linguagem Java foi utilizada conforme estabelecido nos requisitos do projeto, por se tratar de uma linguagem consolidada no ensino de programação orientada a objetos e com amplo suporte a ferramentas de desenvolvimento, testes e interfaces gráficas [IBM 2025]. Além disso foi utilizado o Java SE Development Kit v24.0.1 [Oracle 2025] juntamente com a o ambiente de desenvolvimento *Jetbrains IntelliJ Idea.*

3. Metodologia

3.1. Definição de Requisitos e Funcionalidades

A aplicação *Diário Cultural* foi concebida para permitir ao usuário o registro e acompanhamento de obras culturais consumidas, com foco em livros, filmes e séries. A seguir, são listadas as principais funcionalidades implementadas nesta primeira fase:

- **Cadastro de mídias**: inclusão de informações detalhadas sobre livros, filmes e séries.
- **Avaliação de obras**: pontuação de 1 a 5 estrelas, marcação como visto/lido, data e registro de impressões.
- **Busca por conteúdo**: filtragem por diversos critérios, como título, autor, ano, elenco, entre outros.

- **Listagem com ordenação e filtros**: visualização de conteúdos com base em avaliações, gêneros e datas.

Essas funcionalidades foram definidas com base no enunciado do projeto e serviram como base para a modelagem do sistema.

3.2. Descrição de Alto Nível do Sistema

O sistema foi estruturado em torno de uma hierarquia de classes que representam as mídias (*Livro*, *Filme*, *Série*), com uma classe abstrata *Midia* como base comum. Cada tipo de mídia possui atributos e comportamentos específicos. A classe *Acervo* centraliza o gerenciamento das mídias cadastradas, oferecendo métodos para inserção, busca, listagem e filtragem.

Além disso, classes auxiliares foram desenvolvidas para representar avaliações, *reviews*, temporadas e controle de leitura/visualização. A estrutura busca promover a separação de responsabilidades e a coesão entre componentes.

3.3. Processo de Codificação

A codificação foi conduzida com foco em boas práticas de programação orientada a objetos, utilizando conceitos como encapsulamento, herança, polimorfismo e modularização. A padronização de nomes e a documentação via *Javadoc* foram adotadas desde o início.

Os testes de unidade foram desenvolvidos para validar funcionalidades individuais, como cadastro e avaliação de mídias. Já os testes de integração verificaram o funcionamento conjunto entre as classes do sistema.

3.4. Diagramas

Para auxiliar a modelagem e a implementação do sistema, foram elaborados:

- **Diagrama de Casos de Uso**: apresenta as interações do usuário com o sistema e as funcionalidades disponíveis.
- **Diagrama de Classes**: detalha a estrutura interna do sistema, os relacionamentos entre classes e seus principais métodos e atributos.

Devido ao tamanho dos diagramas, estes foram incluídos no **Apêndice A** deste relatório, com o objetivo de preservar a fluidez da leitura do texto principal sem comprometer o nível de detalhamento técnico necessário.

4. Resultados e Discussões

4.1. Manual de Uso

Na versão atual da aplicação, o uso ocorre via execução diretamente no terminal, utilizando menu que separam as funções como na figura 1. As funcionalidades estão organizadas por meio de métodos acessíveis no código, sendo possível instanciar objetos do tipo `Livro`, `Filme` e `Serie`, além de utilizar a classe `Acervo` para centralizar e manipular os registros.

As principais ações possíveis incluem:

- Cadastro de mídias: criação de objetos com os respectivos atributos.
- Avaliação: uso de métodos como `avaliarLivro`, `avaliarFilme` ou `avaliarTemporada`.
- Busca e listagens: métodos como `buscarPorTitulo`, `buscarPorGenero`, `listarFilmes` e `listarSeries`.

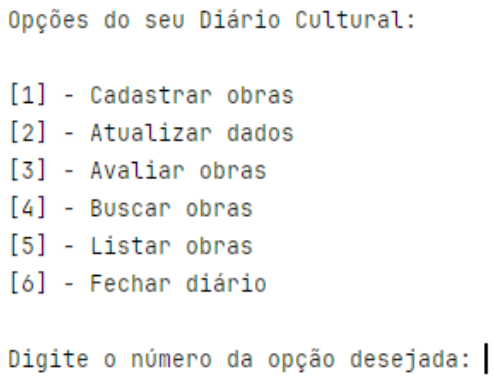

Figure 1. Menu Inicial. Fonte: Autoria Própria

4.2. Dados de Entrada e Saída

Entradas: os dados são fornecidos pelo usuário em tempo de execução, por meio de entradas no console utilizando a classe `Scanner`. São solicitadas informações como título, autor, gênero, ano de lançamento, nota (de 1 a 5), e comentários sobre a obra.

Saídas: os dados cadastrados são apresentados diretamente no console em formato textual, incluindo os resultados de buscas e listagens formatadas como strings, além de médias de avaliações e resumos dos dados cadastrados.

Exemplo de entrada:

```
Livro livro = new Livro("Acotar", "Corte de Espinhos e Rosas", "
   Romance", "2017", "Sarah", "Galera", "978-3-16-148410-0",
   false);
```

Exemplo de saída:

```
Título: Acotar
Título Original: Corte de Espinhos e Rosas
Gênero: Romance
Ano de Lançamento: 2017
Autor: Sarah
Editora: Galera
ISBN: 978-3-16-148410-0
Possui exemplar: Não
Nota: 0
Review: null
Foi lido: Não
```

4.3. Testes Realizados

Foram desenvolvidos testes de unidade para garantir o correto funcionamento das classes individuais, como:

- Cadastro, manipulação e acesso de cada tipo de midia;
- Adição e busca de temporadas em séries;
- Adição e ordenação de midias no acervo.

Testes de integração confirmaram o funcionamento conjunto das classes, como a interação entre `Acervo` e os objetos de mídia. O teste realizado limpa o acervo e adiciona um livro e em seguida testa o processo completo de marcar um livro como visto, avaliar e adicionar uma review.

Todos os testes foram realizados diretamente no ambiente de desenvolvimento, com foco nos principais fluxos de uso.

4.4. Erros e Limitações

Durante a implementação, foram identificadas algumas limitações e oportunidades de melhoria:

- **Validação de entradas do usuário**: como os dados são fornecidos via console utilizando a classe `Scanner`, é necessário implementar verificações mais robustas para garantir que o tipo e o formato das entradas estejam corretos (por exemplo, impedir notas fora da faixa de 1 a 5, datas inválidas ou campos obrigatórios vazios).
- **Tratamento de exceções**: embora existam validações básicas, ainda não foi implementado um tratamento abrangente para exceções como `InputMismatchException` ou `NullPointerException`, o que pode causar encerramentos inesperados do programa em caso de uso incorreto.
- **Persistência de dados**: nesta fase, os dados permanecem apenas em memória durante a execução do programa. A funcionalidade de armazenamento em disco será implementada na **Fase 2** do projeto, permitindo a preservação das informações entre sessões.
- **Interface textual limitada**: por ser uma versão inicial baseada em console, a interação com o usuário ainda é pouco amigável e exige familiaridade com comandos diretos. A futura interface gráfica (Fase 3) visa tornar o uso mais acessível.
- **Padrões de projeto**: até o momento, não foi adotado um padrão de projeto específico. No entanto, boas práticas de organização do código, como a divisão lógica em pacotes, foram seguidas. A aplicação de um padrão formal será considerada durante a refatoração prevista para a **Fase 2**.

Essas limitações não comprometem o funcionamento das funcionalidades básicas da aplicação, mas indicam áreas prioritárias para evolução nas próximas fases.

5. Conclusão

A primeira fase do desenvolvimento da aplicação *Diário Cultural* alcançou os principais objetivos propostos. Foram implementadas as funcionalidades básicas para o cadastro, avaliação e listagem de livros, filmes, séries e temporadas, bem como o controle das informações por meio de uma estrutura orientada a objetos. O sistema foi testado com foco na consistência dos dados e no comportamento conjunto das classes envolvidas.

Apesar da ausência de um padrão de projeto formal, foram aplicadas boas práticas de desenvolvimento, como a divisão em pacotes e a modularização do código, o que contribuiu para a legibilidade e a manutenção da aplicação. A definição de um padrão de projeto será incorporada na próxima etapa, durante a refatoração e o processo de persistência.

Algumas limitações ainda precisam ser abordadas, como o tratamento mais completo de exceções, a persistência dos dados em disco e a melhoria na interação com o usuário. Esses pontos serão contemplados nas Fases 2 e 3 do projeto, que preveem a implementação da camada de armazenamento e da interface gráfica.

Como sugestão de melhoria, destaca-se a possibilidade de permitir múltiplos perfis de usuários, integração com APIs externas para preenchimento automático de dados, e exportação do acervo para formatos como PDF ou CSV. Tais funcionalidades podem ampliar a utilidade da aplicação e enriquecer a experiência do usuário.

References

Aderaldo, C. V. L., de Aquino, C. A. B., and Severiano, M. F. V. (2020). Aceleração, tempo social e cultura do consumo: notas sobre as (im)possibilidades no campo das experiências humanas. *Cadernos EBAPE.BR*, 18(2):365–376.

Henrique, J. (2023). Poo: o que é programação orientada a objetos? Atualizado em 18/09/2023.

IBM (2025). Vantagens do java. Acessado em 25 de abril de 2025.

Oracle (2025). Java downloads. Acessado em 25 de abril de 2025.

A. Apêndice A - Diagramas

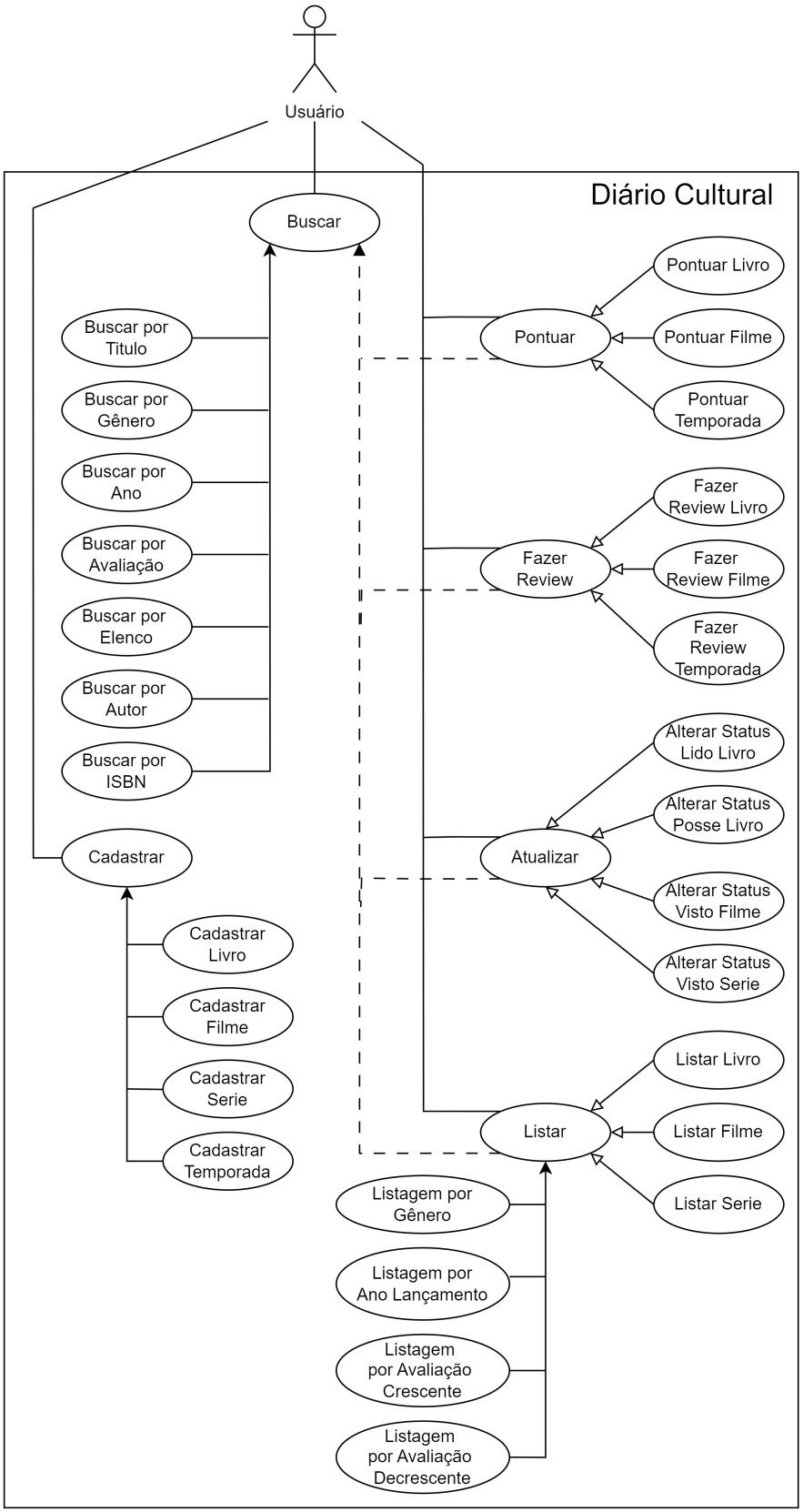

Figure 2. Diagrama de Casos de Uso. Fonte: Autoria Própria

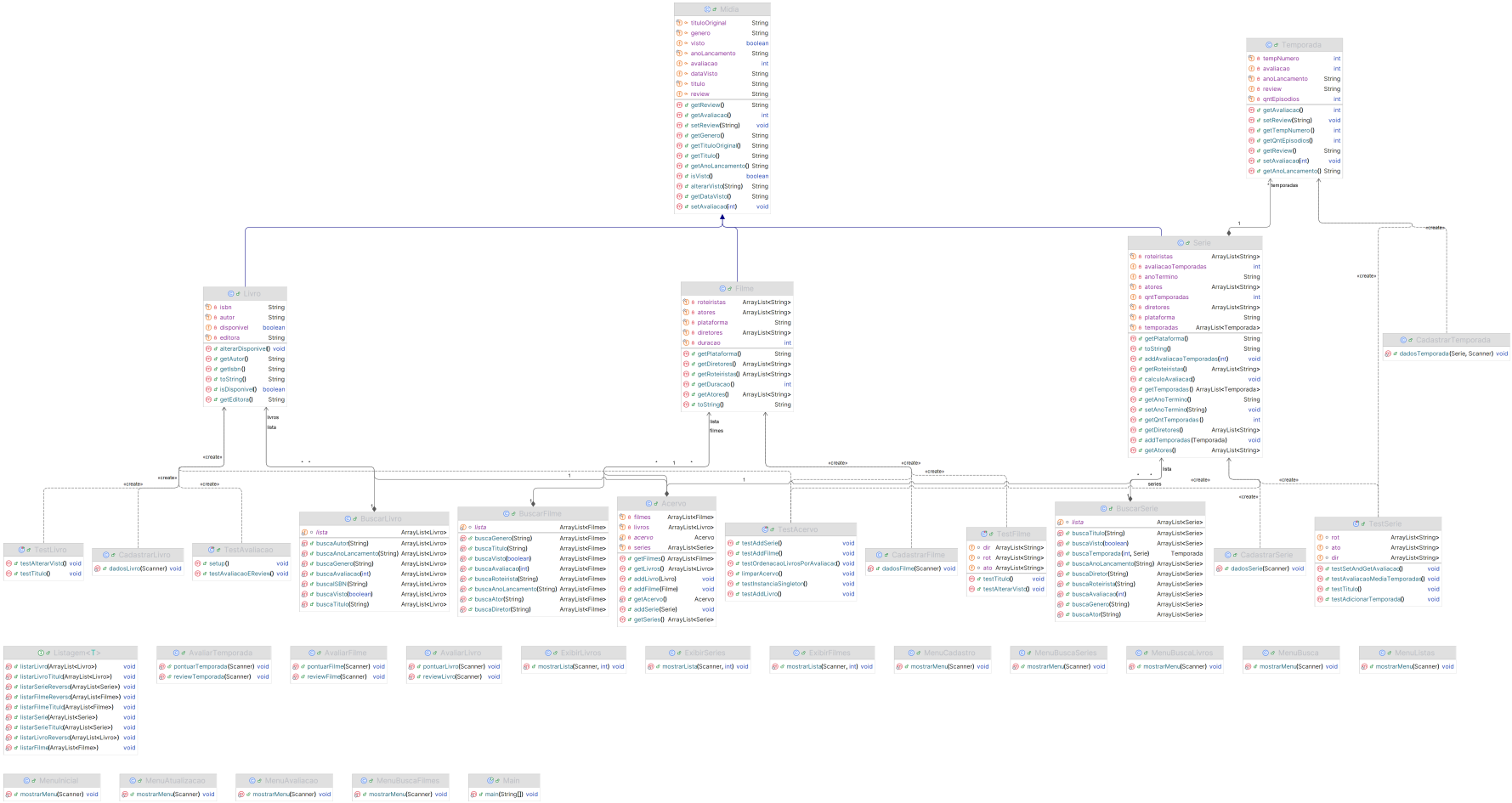

Figure 3. Diagrama de Classes. Fonte: Autoria Própria