**RELATÓRIO TÉCNICO  
Sistema de Gestão de Biblioteca Acadêmica**

# Identificação

|  |  |
| --- | --- |
| Curso: | *Ciencia da computação* |
| Disciplina: | *Programação orientada a objeto* |
| Professor(a): | *Gleidson Porto Batista* |
| Aluno(s)/RGM: | *Filipe Matheus Cabral da Silva - 36100986* |

# 1. Objetivo do Projeto

O objetivo do projeto é desenvolver um sistema de gerenciamento de biblioteca acadêmica, utilizando a linguagem Java e aplicando os principais conceitos da Programação Orientada a Objetos (POO). Este sistema tem como finalidade auxiliar no controle do acervo de livros, cadastro de usuários (alunos e professores), realização de empréstimos e verificação de disponibilidade de livros. O sistema é implementado via interface de console, visando a praticidade no manuseio e entendimento do fluxo lógico por parte do usuário final.

# 2. Funcionalidades Implementadas

- Cadastro de livros: título, autor, ISBN, ano e editora. Classe CadastroLivro  
- Cadastro de usuários: nome, CPF, matrícula, e-mail. Classe abstrata Usuario

- Cadastro de Professor: departamento. Subclasse Herda de Usuario.

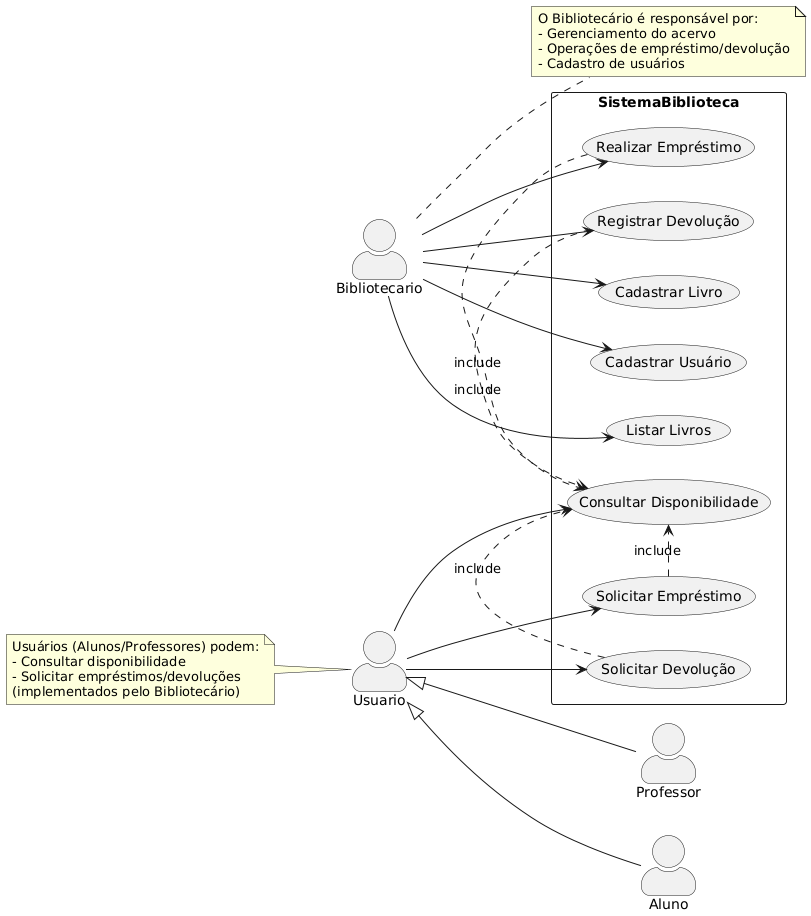
- Cadastro de Aluno: periodo. Subclasse Herda de Usuario.

- Classe Emprestimo. Id, usuario, livro, data e status. O ID é Gerado automatico.  
- Consulta de disponibilidade: verificação dos livros disponíveis para empréstimo.  
- Empréstimos: seleção de livro e usuário, com geração de número de empréstimo único e registro da data.  
- Listagem do acervo: exibição dos livros cadastrados com todas as informações.  
- Validações: impedimento de empréstimo de livros já emprestados, verificação de existência do usuário, entre outros controles básicos.

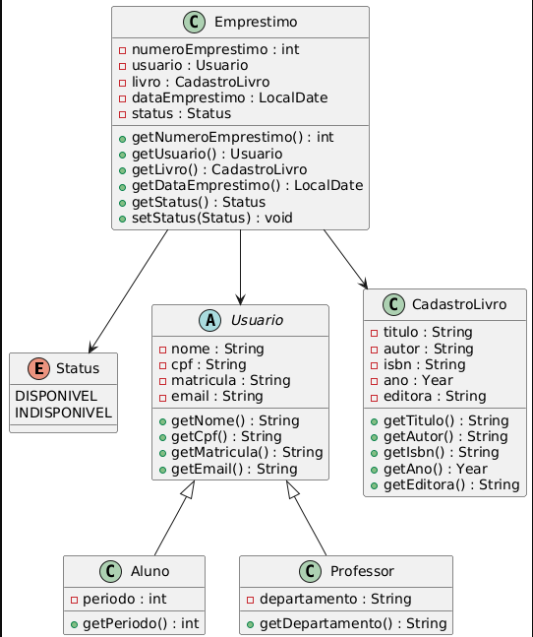
- Devolução: Verifica quem pegou e muda o status.

# 3. Modelagem UML

- Diagrama de Casos de Uso: O ator 'Bibliotecário' interage com o sistema para realizar operações como cadastrar livros e usuários, registrar empréstimos e devoluções, consultar e listar livros. Usuários (alunos e professores) podem consultar disponibilidade e solicitar empréstimos e devoluções, que são processados pelo bibliotecário.

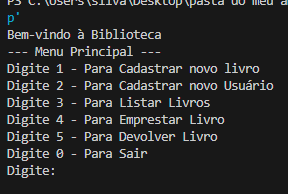


- Diagrama de Classes: Mostra as classes principais do sistema como `Usuario`, `Aluno`, `Professor`, `CadastroLivro`, `Emprestimo` e o enum `Status`. As classes `Aluno` e `Professor` herdam de `Usuario`. A classe `Emprestimo` associa um `Usuario` e um `CadastroLivro`, e seu status é definido pelo enum `Status` (DISPONIVEL, INDISPONIVEL).



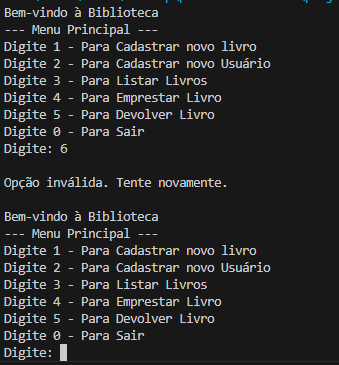
# 4. Interface do Sistema

A interface utilizada no projeto é baseada em linha de comando (console). O sistema apresenta menus interativos ao usuário, que deve digitar números para acessar funcionalidades como cadastrar livros, cadastrar usuários, listar livros e realizar empréstimos. O uso da classe Scanner permite a leitura de dados inseridos pelo teclado. Apesar de ser uma interface simples, é funcional e suficiente para demonstrar a lógica de funcionamento do sistema.

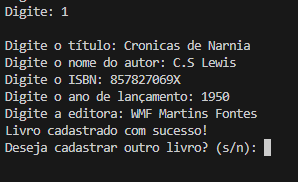
**

*A primeira imagem mostra quando o programa é iniciado, esse é o menu.*

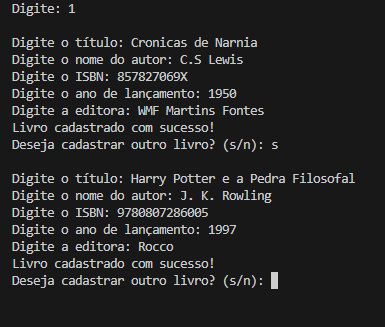
*Nessa parte utilizamos swtch case para o controle do menu, e se o usuario clicar algo diferente tem um controle de exceção para que retorne ao menu novamente e ela escolha a opção certa e não quebre o programa.*

**

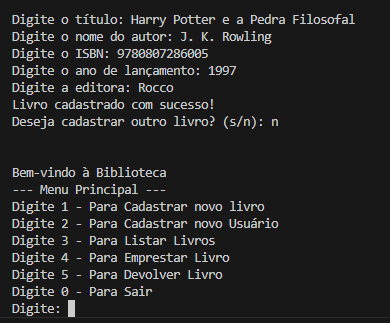
*Nessa tela damos um exemplo colocamos a opção 6 que não esta no Menu e logo ele da a mensagem e retorna novamente.*

**

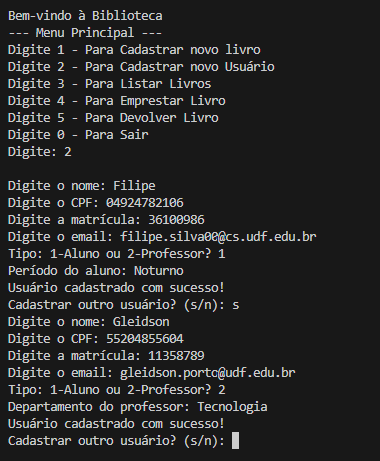
*Seguindo a sequencia e colocando opção 1, Podemos cadastrar o livro, colocamos um laço de repetição que pergunta ao usuario se ele deseja continuar cadastrando livros.*

**

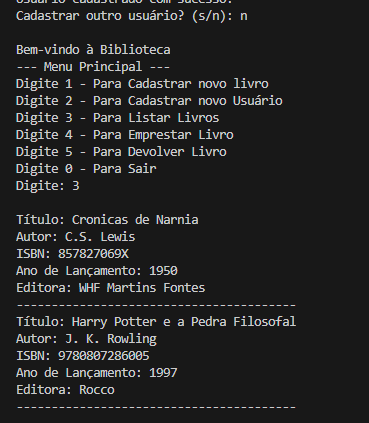
*Se o Usuario digitar s, o programa retorna a pedir os dados e pode cadstra o proximo livro.*

**

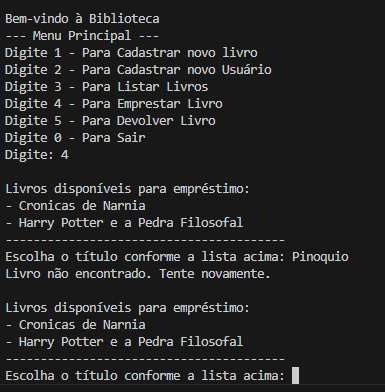
*E se o usuario colocar n, o laço para e volta ao menu inicial.*

**

*No cadastro de usuario é da mesma forma tem um laço de repetição e perguntamos ao usuario se ele deseja cadastrar mais pessoas, alem disso tem no usuario como foi feito subclasses voce escolhe se é um aluno ou professor, para poder instaciar o usuario correto.*

**

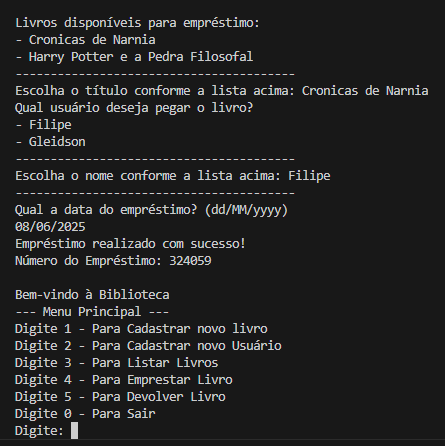
*Quando listamos os livros ele mostra todas as informações de todos os livros cadastrados.*

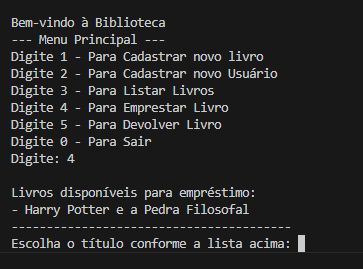
**

*Quando é para emprestar um livro primeiro listamos os disponiveis, caso não seja disponivel nem é mostrado na lista, se o usuario colocar um titulo que não existe na lista o Sistema não aceita e pede o titulo correto.*

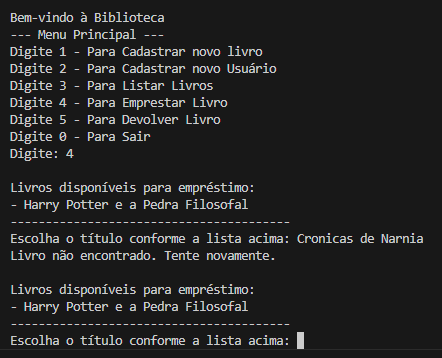
*Escolhendo o titulo correto conforme a imagem abaixo, o Sistema lista os usuarios cadastrados e voce pode escolher, alem disso é dado a data, entao geramos o id aleatorio.*

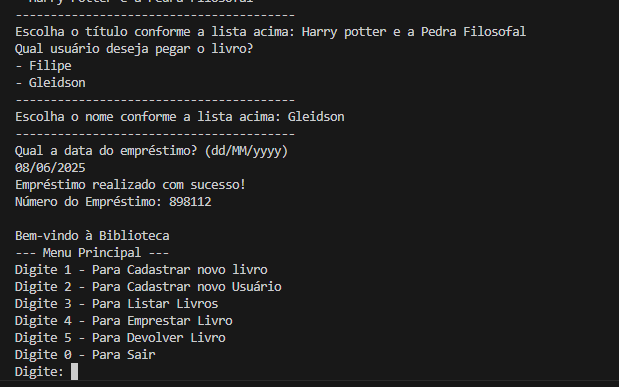
*Existe um controle nessas listagens caso não tenha nenhum usuario ou livro cadastrado tambem.*

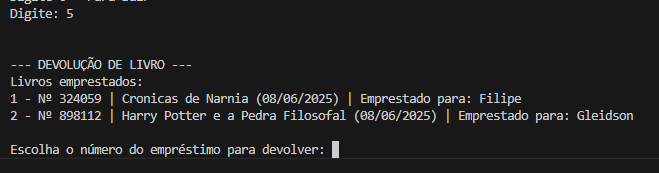
**

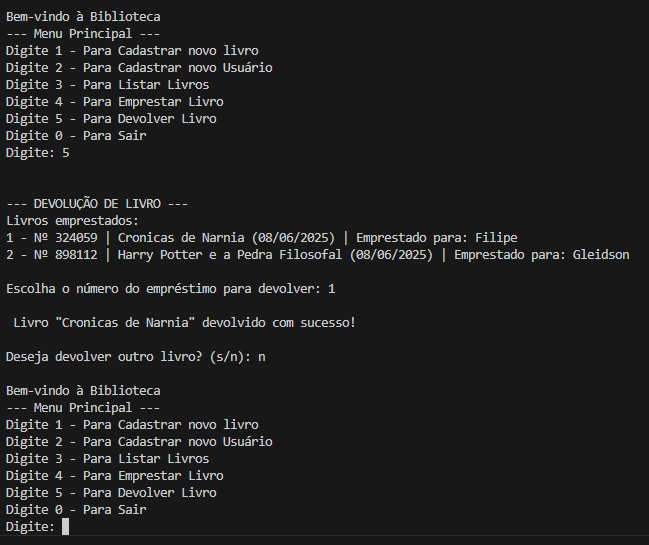
**

*Depois de emprestado se o usuario for no campo emprestimo de novo não aparece mais o livro emprestado, mesmo que ele digite um titulo que tenha na biblioteca.*

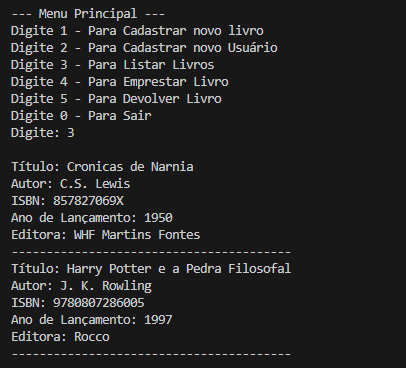
**

**

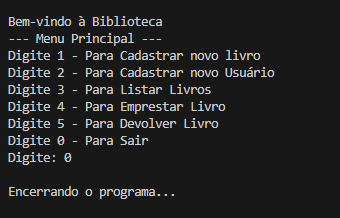
**

**

*A parte de devolução lista os livros emprestados o dia e a pessoa que pegou*

**

*Mesmo se um usuario pegar um livro emprestado na listagem da biblioteca aparece todos os livros.*

**

*Na opção 0 o programa é encerrado.*

*5. Estrutura do Projeto e Código-Fonte*

O sistema está dividido nas seguintes classes e estruturas:  
- Classe principal (App): contém o método main e os menus de navegação.  
- Classe CadastroLivro: representa os livros cadastrados, com atributos como título, autor, ISBN, ano e editora.  
- Classe Usuario: armazena as informações dos usuários como nome, CPF, matrícula, e-mail e tipo.  
- Enum tipo: define se o usuário é ALUNO ou PROFESSOR.  
- Classe Emprestimo: armazena os dados dos empréstimos, como número gerado automaticamente, livro, usuário, data e status.  
- Enum Status: define se o livro está DISPONIVEL ou INDISPONIVEL.  
- Utilização de ArrayList para gerenciar listas de livros, usuários e empréstimos dentro da aplicação.  
- Lógica estruturada com validações (ex: verificação de disponibilidade antes do empréstimo).

# 6. Aplicação dos Conceitos de POO

O projeto aplica os seguintes conceitos fundamentais da Programação Orientada a Objetos:  
- Encapsulamento: todos os atributos são privados e acessados por meio de métodos públicos (getters/setters).  
- Herança: as classes Aluno e Professor herdam de Usuario, que é uma classe abstrata.  
- Polimorfismo: o método abstrato getTipo() é sobrescrito nas subclasses Aluno e Professor.  
- Modularização: separação lógica de responsabilidades entre as classes.  
- Manipulação de coleções: uso de ArrayList para armazenar e manipular dados dinamicamente.  
- Validação de dados: verificação da existência de livros e usuários antes de ações como empréstimo.

# 7. Considerações Finais

O desenvolvimento do sistema de biblioteca acadêmica proporcionou uma rica oportunidade para praticar e consolidar os conhecimentos adquiridos ao longo da disciplina de Programação Orientada a Objetos. A experiência de estruturar as entidades, controlar o fluxo de dados e lidar com listas dinâmicas reforçou a importância da organização e clareza do código. Um desafio comum foi o tratamento correto de entradas de dados pelo teclado e a garantia de que as validações fossem eficazes. Como melhoria futura, seria interessante implementar uma interface gráfica ou a persistência de dados em banco de dados, tornando o sistema mais robusto e aplicável em cenários reais.

# 8. Apêndice

Codigo fonte complete no GitHub: <https://github.com/filipecab/Projeto-POO>