**API Biblioteca - Sistema de Gerenciamento**

API para gerenciamento de biblioteca com controle de usuários, livros e empréstimos, com documentação Swagger/OpenAPI.

Tecnologias Utilizadas

- Java 17

- Spring Boot 3.5.0

- Spring Data JPA

- MySQL 8.0

- Maven

- Hibernate Validator  
- Springdoc OpenAPI 2.2.0 (Documentação da API)

**Documentação Interativa com Swagger**

A API possui documentação completa gerada automaticamente usando o Springdoc OpenAPI:

Acesso à Documentação

- **Swagger UI:** http://localhost:8082/swagger-ui.html

- **OpenAPI JSON:** http://localhost:8082/api-docs

**Recursos Documentados**

1. **Usuários:** CRUD completo de usuários

2. **Livros:** Gestão de acervo com busca e verificação de disponibilidade

3. **Empréstimos:** Operações de empréstimo e devolução de livros

**Configuração Swagger**

A documentação foi configurada através de:

1. Dependência no `pom.xml`:

```xml

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

<version>2.2.0</version>

</dependency>

**Estrutura do Projeto**

src/

├── main/

│ ├── java/

│ │ └── com/

│ │ └── biblioteca/

│ │ ├── config/ # Configuração do Swagger

│ │ ├── controller/ # Controladores REST

│ │ ├── model/ # Entidades JPA

│ │ ├── repository/ # Repositórios de dados

│ │ ├── service/ # Lógica de negócio

│ │ └── Application.java

│ └── resources/

│ └── application.properties

pom.xml

**Configuração do Ambiente**

1. **Pré-requisitos:**

- Java 17 JDK

- MySQL 8.0+

- Maven 3.8+

2. **Banco de Dados:**

```sql

CREATE DATABASE bb\_Api;

**Configuração (application.properties):**

spring.datasource.url=jdbc:mysql://localhost:3306/bb\_Api

spring.datasource.username=root

spring.datasource.password=

server.port=8082

**A aplicação estará disponível em:** <http://localhost:8082/>

Nota: Usar Postman Ou outra Ferramenta de teste de API.

Acesse a documentação interativa em:  
<http://localhost:8082/swagger-ui/index.html>  
  
Conta Do GIT:  
<https://github.com/Inerio29/BibliotecaAPI.git>

**Endpoints da API**

**Usuários (/users)**

|  |  |  |
| --- | --- | --- |
| **Método** | **Endpoint** | **Descrição** |
| GET | /users | Lista todos os usuários |
| GET | /users/{id} | Busca usuário por ID |
| POST | /users | Cria novo usuário |
| PUT | /users/{id} | Atualiza usuário existente |
| DELETE | /users/{id} | Remove usuário |

**Livros (/books)**

|  |  |  |
| --- | --- | --- |
| **Método** | **Endpoint** | **Descrição** |
| GET | /books | Lista todos os livros |
| GET | /books/{id} | Busca livro por ID |
| GET | /books/search/title?title={} | Busca por título |
| GET | /books/search/author?author={} | Busca por autor |
| GET | /books/{id}/available | Verifica disponibilidade |
| POST | /books | Adiciona novo livro |
| PUT | /books/{id} | Atualiza livro existente |
| DELETE | /books/{id} | Remove livro |

**Empréstimos (/loans)**

|  |  |  |
| --- | --- | --- |
| **Método** | **Endpoint** | **Descrição** |
| GET | /loans | Lista todos os empréstimos |
| GET | /loans/{id} | Busca empréstimo por ID |
| POST | /loans/create?userId=X&bookId=Y | Cria novo empréstimo |
| POST | /loans/{loanId}/return | Registra devolução de livro |
| DELETE | /loans/{id} | Remove empréstimo |

**Regras de Negócio**

1. **Validações:**
   * Email único para usuários
   * Campos obrigatórios marcados com @NotBlank
   * Formato de email válido (@Email)
2. **Empréstimos:**
   * Um livro só pode ser emprestado se tiver estoque disponível
   * Um usuário não pode ter dois empréstimos ativos do mesmo livro
   * Ao emprestar livro: stock é decrementado
   * Ao devolver livro: stock é incrementado
3. **Disponibilidade:**
   * Livro disponível quando stock > 0
   * Endpoint /books/{id}/available retorna "Disponível" ou "Indisponível"

**Observações**

* A API segue padrões RESTful
* Tratamento de erros com códigos HTTP apropriados
* Validações de entrada em todos os endpoints críticos
* Configuração de CORS pode ser necessária para integração com frontend