Casos de Teste – Backend (API)

Este documento contém **32 casos de teste** para automação da API do sistema de biblioteca, distribuídos em 7 categorias funcionais.

1. Autenticação e Perfis

Objetivo: Validar autenticação, registro e controle de acesso baseado em perfis.
Total de casos: 4 testes

CT-API-001

Registro de Novo Usuário Aluno (Sucesso)

Endpoint: POST /registro

Objetivo: Validar criação de novo usuário público (sempre Aluno) com dados válidos.

Pré-condições:

- Email `maria.silva@teste.com` não está cadastrado.

Dados de Entrada (JSON):

```
{
  "nome": "Maria Silva",
  "email": "maria.silva@teste.com",
  "senha": "senha123"
}
```

Validações Esperadas:

- Status code: **201**
- Corpo contém campo `mensagem` com `"Usuário criado com sucesso"`
- Corpo contém objeto `usuario` com `id`, `nome`, `email`, `tipo`
- Campo `senha` **não** presente na resposta
- `id` é inteiro positivo
- `tipo` deve ser **1** (Aluno)

CT-API-002

Registro com Email Duplicado (Falha)

Endpoint: POST /registro

Objetivo: Validar que o sistema impede registro com email já existente.

Pré-condições:

- Email `admin@biblioteca.com` já cadastrado.

Dados de Entrada (JSON):

```
{
  "nome": "João Santos",
  "email": "admin@biblioteca.com",
  "senha": "senha456"
}
```

Validações Esperadas:

- Status code: **400**
- Corpo contém `mensagem` = `"Email já cadastrado"`

CT-API-003

Login com Credenciais Válidas (Admin)

Endpoint: POST /login

Objetivo: Validar autenticação bem-sucedida.

Dados de Entrada (JSON):

```
{
  "email": "admin@biblioteca.com",
  "senha": "123456"
}
```

Validações Esperadas:

- Status code: **200**
- Corpo contém `mensagem` = `"Login realizado com sucesso"`
- Corpo contém objeto `usuario` **sem** campo `senha`
- `usuario.tipo` = **3** (Admin)
- Tempo de resposta < **2000 ms**

CT-API-004

Login com Credenciais Inválidas

Endpoint: POST /login

Objetivo: Validar rejeição de credenciais incorretas.

Dados de Entrada (JSON):

```
{
  "email": "admin@biblioteca.com",
  "senha": "senhaerrada"
}
```

Validações Esperadas:

- Status code: **401**
- Corpo contém `mensagem` = `"Email ou senha incorretos"`

2. Livros

Objetivo: Validar operações CRUD de livros, estoque e preço.
Total de casos: 8 testes

CT-API-005

Listar Todos os Livros

Endpoint: GET /livros

Objetivo: Validar retorno da lista completa de livros.

Validações Esperadas:

- Status code: **200**
- Corpo é um **array**
- Cada item possui: `id`, `nome`, `autor`, `paginas`, `descricao`, `imagemUrl`, `dataCadastro`, `estoque`, `preco`
- `paginas` é inteiro positivo
- `dataCadastro` em formato ISO 8601

CT-API-006

Listar Livros Disponíveis

Endpoint: GET /livros/disponiveis

Objetivo: Validar filtragem somente de livros com estoque > 0.

Validações Esperadas:

- Status code: **200**
- Corpo é array
- Todos os itens possuem `estoque > 0`
- Cada item pode possuir campo extra `disponivel = true`

CT-API-007

Buscar Livro por ID (Existente)

Endpoint: GET /livros/1

Objetivo: Validar retorno de livro específico.

Validações Esperadas:

- Status code: **200**
- Corpo contém `id = 1`
- Campos obrigatórios (`nome`, `autor`, `paginas`) não vazios

CT-API-008

Buscar Livro por ID (Inexistente)

Endpoint: GET /livros/9999

Objetivo: Validar tratamento de ID não encontrado.

Validações Esperadas:

- Status code: **404**
- Corpo contém `mensagem = "Livro não encontrado"`

CT-API-009

Adicionar Novo Livro

Endpoint: POST /livros

Objetivo: Validar criação de novo livro com estoque e preço.

Dados de Entrada (JSON):

```
{
  "nome": "Código Limpo",
  "autor": "Robert C. Martin",
  "paginas": 425,
  "descricao": "Manual de boas práticas",
  "imagemUrl": "https://exemplo.com/imagem.jpg",
  "estoque": 10,
  "preco": 59.9
}
```

Validações Esperadas:

- Status code: **201**
- Corpo contém livro criado com `id` gerado automaticamente
- `dataCadastro` preenchido automaticamente (formato ISO)
- Demais campos correspondem aos dados enviados

CT-API-010

Adicionar Livro sem Campos Obrigatórios (Falha)

Endpoint: POST /livros

Objetivo: Validar validação de campos obrigatórios.

Dados de Entrada (JSON):

```
{
  "nome": "",
  "autor": "",
  "paginas": null
}
```

Validações Esperadas:

- Status code: **400**
- Mensagem indicando ausência de campos obrigatórios

CT-API-011

Atualizar Livro Existente

Endpoint: PUT /livros/1

Objetivo: Validar atualização de dados do livro.

Dados de Entrada (JSON):

```
{
  "nome": "Clean Code - Edição Atualizada",
  "autor": "Robert C. Martin",
  "paginas": 464,
  "descricao": "Guia completo atualizado",
  "imagemUrl": "https://exemplo.com/nova-imagem.jpg",
  "estoque": 7,
  "preco": 79.9
}
```

Validações Esperadas:

- Status code: **200**
- `id` permanece **1**
- Campos atualizados com novos valores

CT-API-012

Deletar Livro

Endpoint: DELETE /livros/2

Objetivo: Validar remoção de livro.

Validações Esperadas:

- Status code: **200**
- Mensagem `"Livro removido"`
- `GET /livros/2` subsequente retorna **404**

3. Estatísticas

Objetivo: Validar cálculo correto de estatísticas do sistema.
Total de casos: 1 teste

CT-API-013

Obter Estatísticas da Biblioteca

Endpoint: GET /estatisticas

Objetivo: Validar cálculo de estatísticas gerais.

Validações Esperadas:

- Status code: **200**
- Corpo contém: `totalLivros`, `totalPaginas`, `totalUsuarios`, `usuariosPorTipo`, `livrosDisponiveis`, `arrendamentosPendentes`, `comprasPendentes`
- Todos os valores numéricos são inteiros ≥ 0
- Soma de `usuariosPorTipo.alunos + funcionarios + admins` = `totalUsuarios`

4. Favoritos

Objetivo: Validar funcionalidades de favoritar, remover e listar favoritos.
Total de casos: 4 testes

CT-API-014

Adicionar Livro aos Favoritos

Endpoint: POST /favoritos

Objetivo: Validar funcionalidade de favoritar.

Dados de Entrada (JSON):

```
{
  "usuarioId": 1,
  "livroId": 1
}
```

Validações Esperadas:

- Status code: **201**
- Mensagem `"Livro adicionado aos favoritos"`

CT-API-015

Adicionar Livro Já Favoritado (Falha)

Endpoint: POST /favoritos

Objetivo: Impedir duplicidade de favoritos.

Pré-condições:

- Livro 1 já favoritado pelo usuário 1.

Dados de Entrada (JSON):

```
{
  "usuarioId": 1,
  "livroId": 1
}
```

Validações Esperadas:

- Status code: **400**
- Mensagem indicando que o livro já está nos favoritos

CT-API-016

Listar Favoritos de Usuário

Endpoint: GET /favoritos/1

Objetivo: Validar retorno de favoritos.

Validações Esperadas:

- Status code: **200**
- Corpo é array de livros
- Todos os itens correspondem a livros favoritados pelo usuário 1

CT-API-017

Remover Livro dos Favoritos

Endpoint: DELETE /favoritos

Objetivo: Validar remoção de favorito.

Dados de Entrada (JSON):

```
{
  "usuarioId": 1,
  "livroId": 1
}
```

Validações Esperadas:

- Status code: **200**
- Mensagem `"Livro removido dos favoritos"`

5. Arrendamentos

Objetivo: Validar solicitação, aprovação e rejeição de arrendamentos.
Total de casos: 5 testes

CT-API-018

Criar Arrendamento Válido

Endpoint: POST /arrendamentos

Objetivo: Validar criação de arrendamento com estoque disponível.

Dados de Entrada (JSON):

```
{
  "usuarioId": 3,
  "livroId": 1,
  "dataInicio": "2025-12-20",
  "dataFim": "2025-12-27"
}
```

Validações Esperadas:

- Status code: **201**
- Corpo contém `id`, `usuarioId`, `livroId`, `status`, `criadoEm`
- `status = "PENDENTE"`

CT-API-019

Criar Arrendamento sem Estoque (Falha)

Endpoint: POST /arrendamentos

Objetivo: Validar bloqueio de arrendamento sem estoque.

Pré-condições:

- Livro com `estoque = 0`

Validações Esperadas:

- Status code: **400**
- Mensagem `"Livro sem estoque para arrendamento"`

CT-API-020

Atualizar Status de Arrendamento para APROVADO

Endpoint: PUT /arrendamentos/{id}/status

Objetivo: Validar aprovação e desconto de estoque.

Dados de Entrada (JSON):

```
{
  "status": "APROVADO"
}
```

Validações Esperadas:

- Status code: **200**
- Arrendamento com `status = "APROVADO"`
- Estoque do livro reduzido em 1

CT-API-021

Atualizar Status com Valor Inválido (Falha)

Endpoint: PUT /arrendamentos/{id}/status

Objetivo: Validar rejeição de status inválido.

Dados de Entrada (JSON):

```
{
  "status": "EM_ANALISE"
}
```

Validações Esperadas:

- Status code: **400**
- Mensagem `"Status inválido"`

CT-API-022

Listar Arrendamentos do Usuário

Endpoint: GET /arrendamentos/me?usuarioId=3

Objetivo: Validar listagem de arrendamentos pessoais.

Validações Esperadas:

- Status code: **200**
- Corpo é array
- Todos os itens possuem `usuarioId = 3`

6. Compras

Objetivo: Validar criação, aprovação e cancelamento de compras.
Total de casos: 6 testes

CT-API-023

Criar Compra com Estoque Suficiente

Endpoint: POST /compras

Objetivo: Validar criação de compra válida.

Dados de Entrada (JSON):

```
{
  "usuarioId": 3,
  "livroId": 1,
  "quantidade": 2
}
```

Validações Esperadas:

- Status code: **201**
- Compra com `status = "PENDENTE"`
- Campo `total` = `precoDoLivro * 2`

CT-API-024

Criar Compra com Estoque Insuficiente (Falha)

Endpoint: POST /compras

Objetivo: Validar bloqueio de compra com estoque insuficiente.

Pré-condições:

- Livro com `estoque < 100`

Dados de Entrada (JSON):

```
{
  "usuarioId": 3,
  "livroId": 1,
  "quantidade": 100
}
```

Validações Esperadas:

- Status code: **400**
- Mensagem `"Estoque insuficiente"`

CT-API-025

Aprovar Compra

Endpoint: PUT /compras/{id}/status

Objetivo: Validar aprovação de compra.

Dados de Entrada (JSON):

```
{
  "status": "APROVADA"
}
```

Validações Esperadas:

- Status code: **200**
- Compra com `status = "APROVADA"`
- Estoque do livro reduzido conforme `quantidade`

CT-API-026

Cancelar Compra

Endpoint: PUT /compras/{id}/status

Objetivo: Validar cancelamento de compra.

Dados de Entrada (JSON):

```
{
  "status": "CANCELADA"
}
```

Validações Esperadas:

- Status code: **200**
- Compra com `status = "CANCELADA"`
- Estoque não é alterado (comparado ao valor anterior)

CT-API-027

Listar Compras do Usuário

Endpoint: GET /compras/me?usuarioId=3

Objetivo: Validar listagem de compras pessoais.

Validações Esperadas:

- Status code: **200**
- Array de compras do usuário 3

CT-API-028

Listar Todas as Compras

Endpoint: GET /compras

Objetivo: Validar listagem de todas as compras (admin).

Validações Esperadas:

- Status code: **200**
- Array com compras de múltiplos usuários (quando existirem)

7. Admin Usuários

Objetivo: Validar operações CRUD de usuários por administrador.
Total de casos: 4 testes

CT-API-029

Listar Usuários

Endpoint: GET /usuarios

Objetivo: Validar listagem de usuários.

Validações Esperadas:

- Status code: **200**
- Array de usuários sem campo `senha`

CT-API-030

Atualizar Usuário

Endpoint: PUT /usuarios/2

Objetivo: Validar atualização de dados do usuário.

Dados de Entrada (JSON):

```
{
  "nome": "João Funcionário Atualizado",
  "email": "func.atualizado@biblio.com",
  "tipo": 2
}
```

Validações Esperadas:

- Status code: **200**
- Usuário 2 com novos `nome` e `email`
- `tipo = 2`

CT-API-031

Excluir Usuário (Não-Admin Principal)

Endpoint: DELETE /usuarios/3

Objetivo: Validar exclusão de usuário comum.

Validações Esperadas:

- Status code: **200**
- Mensagem `"Usuário deletado com sucesso"`

CT-API-032

Tentar Excluir Admin Principal (Falha)

Endpoint: DELETE /usuarios/1

Objetivo: Validar proteção do admin principal.

Validações Esperadas:

- Status code: **403**
- Mensagem indicando que o admin principal não pode ser deletado