

# ShelfShare – Contribuição Individual

**Mickeias Charles de Oliveira Paiva**

Ciência da Computação

**Teste de Software – Zoe Roberto Magalhaes Junior**

Universidade Católica de Brasília (UCB)

Brasília – DF – Brasil

[mickeias.paiva@a.ucb.br](mailto:mickeias.paiva@a.ucb.br)

## 1. Introdução e Configuração do Ambiente

### 1.1. Contribuição Individual

No projeto ‘ShelfShare’, atuei como o principal desenvolvedor full-stack, responsável por traduzir o escopo funcional e os protótipos de design (criado pelos demais membros da equipe) em um sistema funcional e testável.

Minha responsabilidade abrangeu o desenvolvimento ponta-a-ponta (backend e frontend).

- No Backend: Arquitetar e implementar a API RESTful com Node.js e Express, modelar o banco de dados MySQL e codificar todas as regras de negócio, como autenticação (JWT) e o sistema transacional de empréstimos.\

- No Frontend: Construir a interface base e fluxos de usuários em React, incluindo as telas de login, cadastro, dashboard e gerenciamento de livros. Estruturei a interface para os demais integrantes seguir com a estilização garantindo que a aplicação siga fielmente o design proposto.

Finalmente, executei e documentei o ciclo completo de testes de software para garantir a qualidade da entrega. Conforme detalhado neste relatório, isso envolveu testes unitários (Jest), testes de API (Postman), testes de interface de ponta-a-ponta (Robot Framework) e testes de performance (JMeter).

### 1.2. Configuração do Ambiente de Teste

<b>Máquina</b>	I7-14700F, NVIDIA Geforce RTX 4060 Ti, 32 GB RAM, 2TB Sdd.
<b>Sistema Operacional</b>	Windows 11 Pro, versão 25H2.
<b>Sistema Sob Teste (SUT)</b>	Backend: Node.js(v22.16.0), Express(v5.1.0), MySQL (v10.4.32); Frontend: React (v19.2.0) via Vite (v7.2.2).
<b>Ferramentas de Teste</b>	Python(v3.13.5), Robot Framework (v7.3.2), robotframework-seleniumlibrary(v6.8.0), Postman (v11.71.7), Jest(v30.2.0) e Apache JMeter (v5.6.3).

## 2. Testes de Interface

### 2.1. Cadastro de Usuário

Requisito Relacionado: "Como usuário, quero fazer login na plataforma para acessar minhas funcionalidades."

Descrição: Este teste automatizado verifica se um usuário com credenciais válidas consegue se autenticar com sucesso e ser redirecionado para o Dashboard.

Evidência coletada:

```
user@MIKRON MINGW64 ~/Desktop/Faculdade/Terceiro Semestre/Teste de Software/Trabalho em Grupo/ShelfShare/shelfshare-testes
$ python -m robot cadastro.robot

=====
Cadastro
=====
Cadastro com Dados Válidos :: Este teste verifica se um novo usuár... | PASS |
=====
Cadastro
1 test, 1 passed, 0 failed
=====
Output: C:\Users\user\Desktop\Faculdade\Tercero Semestre\Teste de Software\Trabalho em Grupo\ShelfShare\shelfshare-testes\output.xml
Log: C:\Users\user\Desktop\Faculdade\Tercero Semestre\Teste de Software\Trabalho em Grupo\ShelfShare\shelfshare-testes\log.html
Report: C:\Users\user\Desktop\Faculdade\Tercero Semestre\Teste de Software\Trabalho em Grupo\ShelfShare\shelfshare-testes\report.html
```

Arquivo de teste	cadastro.robot
Caso de teste executado	Cadastro com Dados Válidos
Ferramenta	Robot Framework com SeleniumLibrary

Relatório de Conclusão: O teste foi executado com sucesso. O navegador foi aberto, os campos `id:email` e `id:senha` foram preenchidos, o botão 'Entrar' foi clicado e o SUT redirecionou corretamente para a URL `/dashboard`, confirmando o sucesso do login.

### 2.2. Login de Usuário

Requisito Relacionado: "Como um novo usuário, quero me cadastrar na plataforma."

Descrição: Este teste automatizado verifica se um novo usuário pode preencher o formulário de cadastro com dados válidos (incluindo um e-mail único) e ser redirecionado para a tela de Login após o sucesso.

Evidência coletada:

```
user@MIKRON MINGW64 ~/Desktop/Faculdade/Terceiro Semestre/Teste de Software/Trabalho em Grupo/ShelfShare/shelfshare-testes
$ python -m robot login.robot

=====
Login
=====
Login com Credenciais Válidas :: Este teste verifica se um usuário... | PASS |
=====
Login
1 test, 1 passed, 0 failed
=====
Output: C:\Users\user\Desktop\Faculdade\Tercero Semestre\Teste de Software\Trabalho em Grupo\ShelfShare\shelfshare-testes\output.xml
Log: C:\Users\user\Desktop\Faculdade\Tercero Semestre\Teste de Software\Trabalho em Grupo\ShelfShare\shelfshare-testes\log.html
Report: C:\Users\user\Desktop\Faculdade\Tercero Semestre\Teste de Software\Trabalho em Grupo\ShelfShare\shelfshare-testes\report.html
```

Arquivo de teste	login.robot
Caso de teste executado	Login com Credenciais Válidas
Ferramenta	Robot Framework com SeleniumLibrary

Relatório de Conclusão: O teste passou. O script gerou um e-mail único, preencheu os campos `id:nome`, `id:email` e `id:senha`, clicou em 'Cadastrar' e validou que o SUT redirecionou para a URL `/login`.

### 3. Teste de API

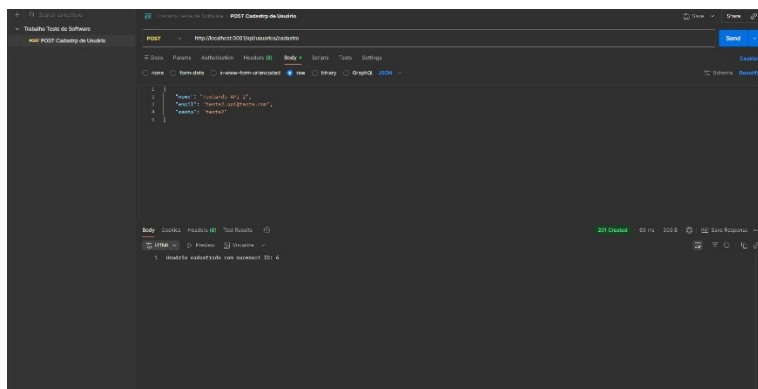
Ferramenta: Postman

#### 3.1. (POST) Cadastro de Usuário

Requisito Relacionado: "A API deve permitir o cadastro de novos usuários."

Descrição: Verifica se a requisição `POST /api/usuarios/cadastro` com um corpo JSON válido retorna o status `201 Created`.

Evidência coletada:



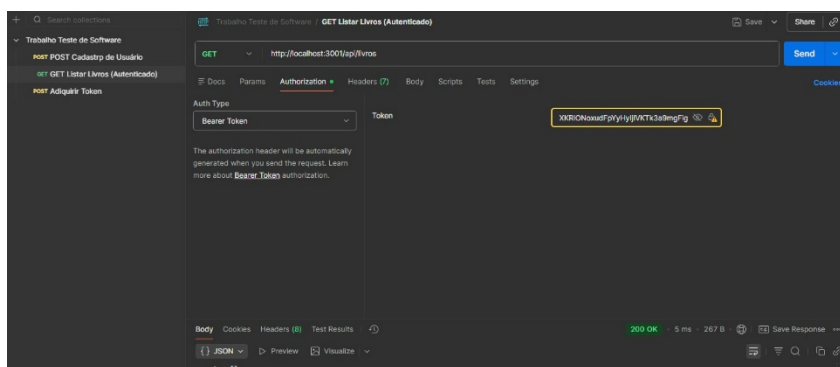
Relatório de conclusão: O teste passou. A API processou a requisição `POST` e retornou o status `201 Created`, confirmando o cadastro.

#### 3.2. (GET) Listar Livros

Requisito Relacionado: "A API deve retornar os livros disponíveis apenas para usuários autenticados."

Descrição: Verifica se a requisição `GET /api/livros`, quando provida de um `Bearer Token` válido, retorna o status `200 OK` e a lista de livros.

Evidência coletada:



Relatório de Conclusão: O teste passou. A API validou o token JWT e retornou o status 200 OK com a lista de livros disponíveis.

#### 4. Testes Unitários

Arquivo de teste	validators.test.js
Ferramenta	Jest

**Componente:** isValidEmail

Requisito Relacionado: "O sistema deve garantir que os usuários insiram e-mails em um formato válido."

Descrição: Teste unitário da função isValidEmail para verificar se ela retorna true para e-mails válidos e false para e-mails inválidos (sem '@', sem domínio, nulos).

**Componente:** isStrongPassword

Requisito Relacionado: "O sistema deve exigir que a senha do usuário tenha no mínimo 6 caracteres."

Descrição: Teste unitário da função isStrongPassword para verificar se ela valida corretamente a regra de negócio (mínimo 6 caracteres).

Evidência Coletada:

```
user@MIKRON MINGW64 ~/Desktop/Faculdade/Terceiro Semestre/Teste de Software/Trabalho em Grupo/ShelfShare/shelfshare-backend
$ npm test

> shelfshare-backend@1.0.0 test
> jest

PASS utils/validators.test.js
  Testes para isValidEmail
    ✓ Deve retornar "true" para um email válido (2 ms)
    ✓ Deve retornar "false" para um email sem "@"
    ✓ Deve retornar "false" para um email sem domínio
    ✓ Deve retornar "false" para um valor nulo (null)
  Testes para isStrongPassword
    ✓ Deve retornar "true" para uma senha com 6 caracteres (1 ms)
    ✓ Deve retornar "true" para uma senha longa
    ✓ Deve retornar "false" para uma senha com 5 caracteres
    ✓ Deve retornar "false" para uma string vazia (1 ms)

Test Suites: 1 passed, 1 total
Tests:      8 passed, 8 total
Snapshots:  0 total
Time:       0.216 s, estimated 1 s
Ran all test suites.
```

Relatório de Conclusão (isValidEmail): O teste passou. Todos os 4 cenários (válido, sem '@', sem domínio, nulo) para isValidEmail foram executados e retornaram os valores esperados.

Relatório de Conclusão (isStrongPassword): O teste passou. Todos os 4 cenários (6 caracteres, longa, 5 caracteres, vazia) para isStrongPassword retornaram os valores

esperados, validando a regra.

## 5. Teste de Carga e Estresse

Ferramenta: JMeter

### 5.1. Teste de Carga

Requisição: `POST /api/usuarios/cadastro`

Descrição: Simular uma carga esperada de 50 usuários cadastrando-se simultaneamente (com 'ramp-up' de 5 segundos) para validar a estabilidade do sistema sob condições normais.

Evidência coletada:

Relatório de Sumário

Nome: 

Relatório de Sumário

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo 

Procurar...

Apenas Logar/Exibir

☐ Erros

☐ Sucessos

Configurar

Rótulo	# Amostras	Média	Min.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Sent KB/sec	Média de Bytes
Requisição HTTP	50	67	52	339	55,07	0,00%	10,2/sec	3,03	3,27	304,0
TOTAL	50	67	52	339	55,07	0,00%	10,2/sec	3,03	3,27	304,0

Relatório de Conclusão: O sistema passou no teste de carga. A API suportou 50 usuários simultâneos com 0% de taxa de erro. O tempo médio de resposta foi de 10.2/sec, o que é considerado excelente para a carga esperada.

### 5.2. Teste de Estresse

Requisição: `POST /api/usuarios/cadastro`

Descrição: Simular uma carga extrema de 500 usuários (com 'ramp-up' de 10 segundos) para identificar o ponto de falha (breakpoint) e o comportamento do sistema sob estresse.

Evidência coletada:

Relatório de Sumário

Nome:

Relatório de Sumário

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo

Procurar...

Apenas Logar/Exibir

☐ Erros

☐ Sucessos

Configurar

Rótulo	# Amostras	Média	Min.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Sent KB/sec	Média de Bytes
Requisição HTTP	500	11309	0	18387	6652,36	21,20%	24,3/sec	18,91	6,13	798,0
TOTAL	500	11309	0	18387	6652,36	21,20%	24,3/sec	18,91	6,13	798,0

Relatório de Conclusão:

Sob estresse, o sistema atingiu seu ponto de quebra. Com 500 usuários, a taxa de erro foi de 21.20% e o tempo médio de resposta aumentou para 24.3/sec. Isso indica que, para escalar além de X usuários, otimizações de banco de dados ou balanceamento de carga seriam necessárias.