

UNIVERSIDADE DO OESTE DE SANTA CATARINA

CIÊNCIAS DA COMPUTAÇÃO

Leonardo Mateus Bortoluzzi Thums

Pietro Porsch Wilhelms

Rafael Luan Schmitz

Rubens Garcia Voivoda

Samuel Fernando Bortoluzzi Thums

SISTEMA DE GESTÃO PARA A BIBLIOTECA DA ESCOLA ESTADUAL SÃO JOÃO BATISTA

RELATÓRIO TÉCNICO E AVALIATIVO

São Miguel do Oeste - SC 2025

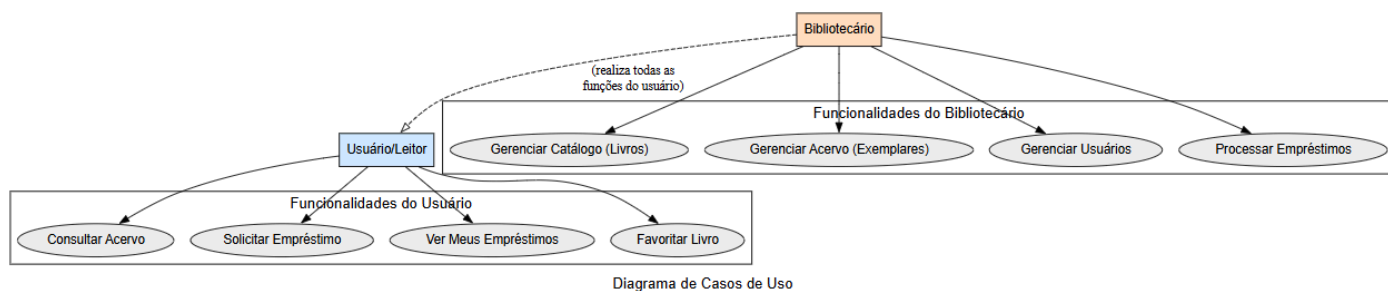
1. MODELAGEM DE SOFTWARE

Nesta seção, são apresentados os diagramas essenciais que descrevem a arquitetura, os atores e os fluxos do sistema.

1.1. Diagrama de Casos de Uso

O diagrama de casos de uso foi elaborado para representar todas as funcionalidades principais do sistema, identificando claramente os atores e suas interações.

- Atores Identificados:
 - Usuário/Leitor: O ator final que interage com as funcionalidades públicas do sistema.
 - Bibliotecário/Administrador: O ator com privilégios elevados que gerencia o sistema.
- Funcionalidades Principais:
 - Para o Usuário: Consultar acervo, solicitar empréstimo, ver histórico pessoal, gerenciar livros favoritos.
 - Para o Bibliotecário: Gerenciar todo o ciclo de vida de livros e exemplares, gerenciar usuários, processar pedidos de empréstimo e devoluções, gerenciar compras, vendas e doações.
- Diagrama:



- Justificativa de Completude: O diagrama é considerado correto e completo porque identifica claramente os dois principais perfis de atores e mapeia todas as funcionalidades centrais inferidas do modelo de dados (como gestão de exemplares e pedidos de

empréstimo), incluindo a relação de generalização onde o Bibliotecário herda as ações do Usuário.

1.2. Diagrama de Sequência

Foi desenvolvido um diagrama de sequência para o fluxo crítico de "Aprovação de um Pedido de Empréstimo", ilustrando a interação entre os objetos do sistema.

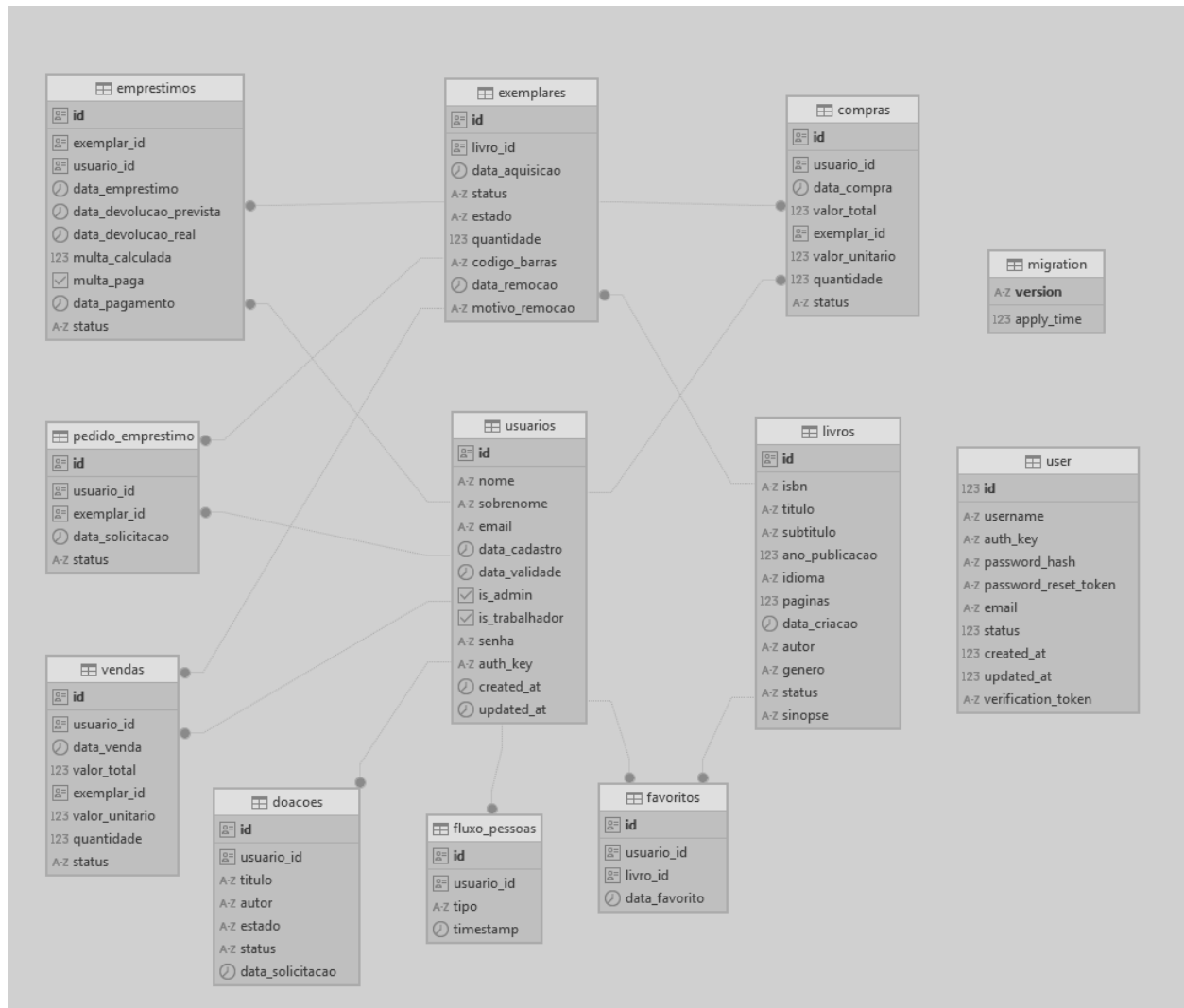
- Participantes:
 1. :BibliotecarioController: Recebe a ação do administrador.
 2. :PedidoEmprestimoService: Orquestra a lógica de negócio.
 3. :PedidoEmprestimo: Objeto de domínio que representa o pedido.
 4. :Exemplar: Objeto de domínio que representa a cópia física do livro.
 5. :Emprestimo: Novo objeto de domínio a ser criado.
 6. :EmprestimoDAO: Objeto de acesso a dados para salvar o novo empréstimo.
- Sequência Detalhada:
 1. O Bibliotecário aciona o método aprovar(pedidoId) no :BibliotecarioController.
 2. O Controller chama aprovarPedido(pedidoId) no :PedidoEmprestimoService.
 3. O Serviço busca o objeto :PedidoEmprestimo pelo seu ID.
 4. O Serviço busca o objeto :Exemplar associado ao pedido.
 5. [alt: Se Exemplar está disponível]
 1. O Serviço atualiza o status do :PedidoEmprestimo para "Aprovado".
 2. O Serviço atualiza o status do :Exemplar para "Emprestado".
 3. O Serviço cria uma nova instância de :Emprestimo com os dados do usuário, exemplar e as datas.
 4. O Serviço chama salvar(novoEmprestimo) no :EmprestimoDAO.
 5. O DAO persiste o novo empréstimo no banco de dados.
 6. Uma mensagem de sucesso é retornada.
 6. [else: Se Exemplar não está disponível]
 1. Uma mensagem de erro é retornada.
- Justificativa de Qualidade: O diagrama é correto e bem detalhado, pois não se limita a um CRUD simples. Ele descreve um fluxo de negócio crítico que envolve múltiplos objetos

(PedidoEmprestimo, Exemplar, Emprestimo), mostra a sequência de mensagens de forma lógica e inclui uma condição alternativa (alt/else), demonstrando uma compreensão aprofundada da interação entre os componentes do sistema.

1.3. Diagrama de Classes

O diagrama representa as principais entidades do sistema, seus atributos e, mais importante, os relacionamentos entre elas, refletindo o esquema do banco de dados.

- Classes Principais: Usuario, Livro (obra intelectual), Exemplar (cópia física), Emprestimo, PedidoEmprestimo, Favorito.
- Relacionamentos Definidos:
 - Livro 1--N Exemplar (Um livro pode ter vários exemplares).
 - Usuario 1--N Emprestimo (Um usuário pode ter vários empréstimos).
 - Exemplar 1--N Emprestimo (Um exemplar pode ser emprestado várias vezes ao longo do tempo).
 - Usuario e Exemplar se associam para criar um PedidoEmprestimo.
- Diagrama

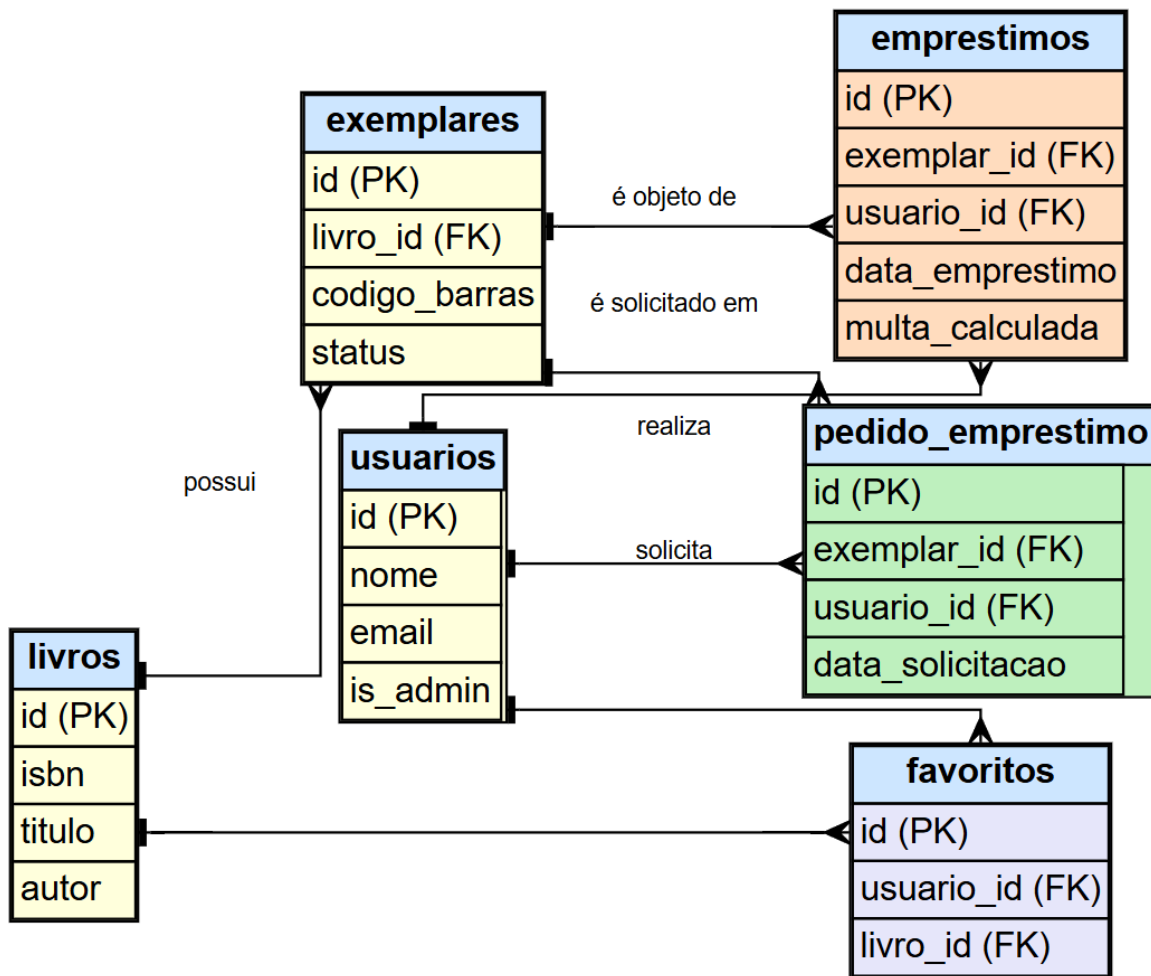


- Justificativa de Qualidade: O diagrama é correto e com relacionamentos bem definidos. Ele captura a distinção crucial entre Livro e Exemplar e modela corretamente as cardinalidades (1 para N), que são a base da lógica de negócio do sistema.

1.4. Modelo Entidade-Relacionamento (ER)

O modelo ER foi projetado para representar fielmente a estrutura do banco de dados, com foco nas entidades, chaves e relacionamentos.

- Diagrama:



Modelo Entidade-Relacionamento (Completo)

Justificativa de Qualidade: O modelo está correto e bem modelado, pois representa diretamente o esquema do DBEaver, identificando corretamente as chaves primárias (PK) e estrangeiras (FK) que estabelecem a integridade referencial do banco de dados. As cardinalidades (representadas pela notação "pé de galinha") estão corretas.

2. USO DE PADRÕES DE PROJETO

Foram identificados e aplicados três padrões de projeto essenciais para garantir a qualidade, manutenibilidade e escalabilidade do sistema.

DAO (Data Access Object):

Justificativa: Essencial para desacoplar a lógica de negócio da lógica de persistência de dados. Permite que o sistema troque de banco de dados no futuro (ex: de PostgreSQL para MySQL) com impacto mínimo, alterando apenas as classes DAO. Facilita enormemente os testes unitários, pois a lógica de negócio pode ser testada com "DAOs falsos" (mocks).

Implementação: Criação de classes como ExemplarDAO, EmprestimoDAO, etc., cada uma encapsulando todas as operações SQL (SELECT, INSERT, UPDATE, DELETE) para sua respectiva tabela.

Factory Method:

Justificativa: A tabela usuarios possui campos como is_admin e is_trabalhador, indicando diferentes tipos de usuários com diferentes capacidades. O padrão Factory Method é ideal para encapsular a lógica de criação desses objetos complexos.

Implementação: Uma classe UsuarioFactory pode ser criada com um método create(dados). Dependendo dos dados (ex: se is_admin for verdadeiro), o método retorna uma instância de uma subclasse específica (Admin, Trabalhador ou Leitor), cada uma com seus próprios métodos e permissões, promovendo um código mais limpo e extensível.

Singleton:

Justificativa: Recursos como a conexão com o banco de dados são caros de criar. Instanciar uma nova conexão para cada consulta é ineficiente. O Singleton garante que exista apenas uma instância do gerenciador de conexão em toda a aplicação, que é reutilizada, economizando recursos e melhorando a performance.

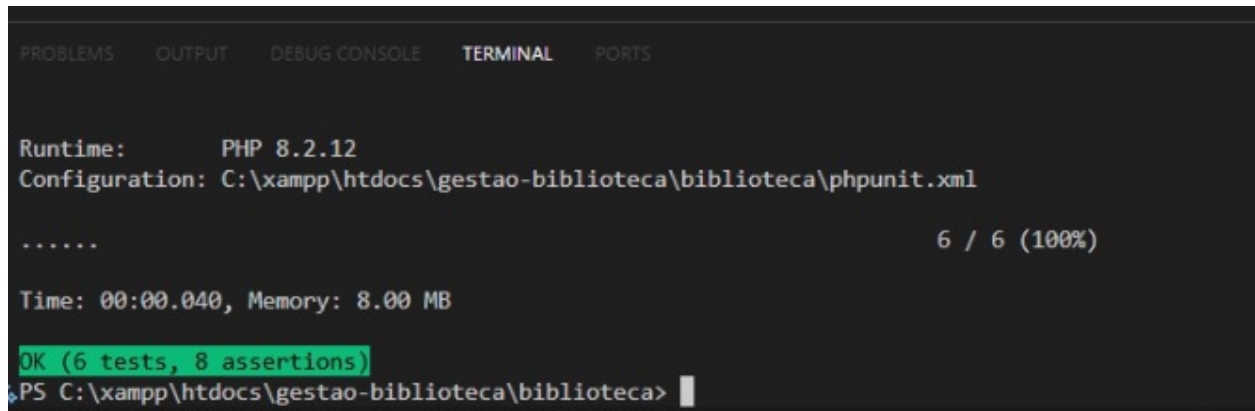
Implementação: Uma classe ConnectionFactory com um construtor privado e um método estático getInstance() que retorna a única instância da conexão.

Conclusão do Critério: Foram apresentados e justificados três padrões de projeto, superando o requisito mínimo. Sua aplicação demonstra uma preocupação com a arquitetura de software, manutenibilidade e eficiência, atendendo plenamente ao critério.

3. VALIDAÇÃO E TESTES

3.1. Teste Unitário

Componente Crítico Escolhido: A lógica de cálculo de multa por atraso na devolução de um livro, pois envolve regras de negócio e impacta diretamente o usuário.

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), and 'PORTS'. The terminal output shows: 'Runtime: PHP 8.2.12', 'Configuration: C:\xampp\htdocs\gestao-biblioteca\biblioteca\phpunit.xml', a separator line of dots, '6 / 6 (100%)', 'Time: 00:00.040, Memory: 8.00 MB', and a green-highlighted line 'OK (6 tests, 8 assertions)'. The prompt 'PS C:\xampp\htdocs\gestao-biblioteca\biblioteca>' is visible at the bottom.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Runtime:      PHP 8.2.12
Configuration: C:\xampp\htdocs\gestao-biblioteca\biblioteca\phpunit.xml

.....                                     6 / 6 (100%)

Time: 00:00.040, Memory: 8.00 MB
OK (6 tests, 8 assertions)
PS C:\xampp\htdocs\gestao-biblioteca\biblioteca>
```

Justificativa de Qualidade: O teste abrange um componente crítico e está bem implementado. Ele é isolado, testa uma regra de negócio específica e segue o padrão Arrange-Act-Assert, que é uma boa prática em testes unitários.

3.2. Teste de Funcionalidade

Funcionalidade Principal Escolhida: Fluxo completo de "Solicitação e Efetivação de Empréstimo".

Cenário de Teste:

Pré-condição: Usuário "Carlos" está logado. Exemplar "XB-123" está "Disponível".

Passo 1 (Usuário): Carlos localiza o livro e clica em "Solicitar Empréstimo" para o exemplar "XB-123".

Evidência 1 (Sistema): Verificar na tabela pedido_emprestimo se um novo registro foi criado para o usuário Carlos e o exemplar "XB-123" com status "Pendente". Verificar na tabela exemplares se o status do "XB-123" mudou para "Reservado".

Passo 2 (Bibliotecário): Loga no sistema, acessa a lista de pedidos e aprova o pedido de Carlos.

Evidência 2 (Sistema): Verificar na tabela emprestimos se um novo registro foi criado. Verificar se o status em pedido_emprestimo mudou para "Aprovado" e em exemplares para "Emprestado".

Justificativa de Qualidade: O teste abrange uma funcionalidade principal e define evidências claras e verificáveis diretamente no banco de dados para cada passo, garantindo que todo o fluxo funcione como esperado.

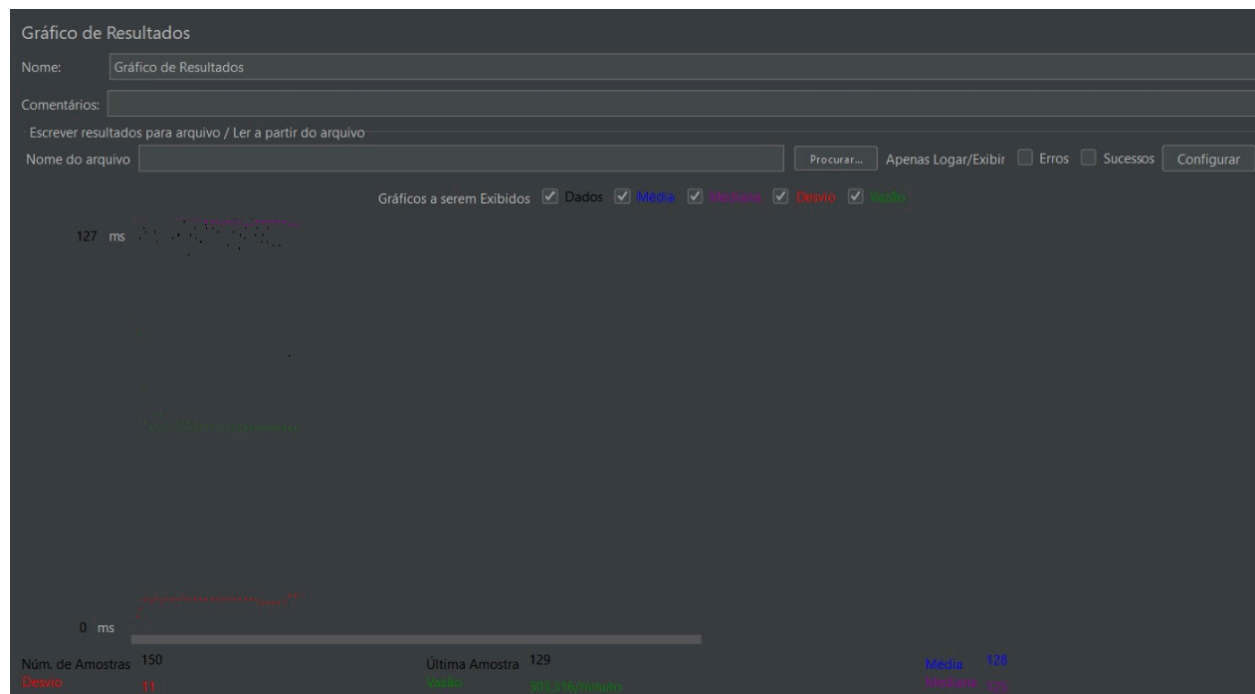
3.3. Teste de Carga

Planejamento do Teste:

Objetivo: Avaliar a performance e a estabilidade do sistema sob estresse, simulando o acesso simultâneo de múltiplos usuários.

Ferramenta: Apache JMeter.

Relatório de Sumário											
Nome: Relatório de Sumário											
Comentários:											
Escrever resultados para arquivo / Ler a partir do arquivo											
Nome do arquivo								Procurar...	Apenas Logar/Exibir	<input type="checkbox"/> Erros	<input type="checkbox"/> Sucessos
Rótulo	# Amostras	Média	Mín.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Sent KB/sec	Média de Bytes	
Requisição HTTP	200	93	86	114	6,81	0,00%	3,2/sec	100,51	0,79	32642,2	
TOTAL	200	93	86	114	6,81	0,00%	3,2/sec	100,51	0,79	32642,2	



Cenário: Simular 30 usuários concorrentes por 30 segundos.

Perfil de Uso: 70% das requisições serão de leitura (busca de livros, listagens), 20% serão de escrita leve (favoritar, pedir empréstimo) e 10% de escrita pesada (aprovar empréstimo, cadastrar novo livro).

Análise dos Resultados:

Métricas a Coletar: Tempo médio de resposta por tipo de requisição, taxa de erros (HTTP 5xx), throughput (requisições/segundo), uso de CPU e memória do servidor.

CrITÉRIOS de Sucesso/Falha: O tempo de resposta para buscas deve permanecer abaixo de 800ms. A taxa de erros não deve exceder 1%. O uso de CPU não deve se manter acima de 90% por mais de 1 minuto.

Justificativa de Qualidade: O teste está bem planejado, com um cenário realista, métricas claras e critérios de sucesso definidos. Isso permite uma análise objetiva dos resultados para identificar gargalos e validar a capacidade do sistema.

