UNIVERSIDADE CATÓLICA DE BRASÍLIA

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

NOME DOS AUTORES

SISTEMA DE GESTÃO DE LIVRARIA

TAGUATINGA-DF

2025

NOME DOS AUTORES

SISTEMA DE GESTÃO DE LIVRARIA

Este trabalho tem como objetivo desenvolver um sistema completo de gestão para livraria utilizando uma arquitetura híbrida que integra MySQL e MongoDB. A proposta demonstra na prática como bancos relacionais e NoSQL podem trabalhar em conjunto.

Orientador: Prof. Jefferson Salomão Rodrigues

TAGUATINGA-DF

2025

RESUMO

Este artigo apresenta o desenvolvimento de um sistema de gestão para livraria utilizando uma arquitetura híbrida que combina MySQL como banco de dados relacional principal e MongoDB para funcionalidades específicas. O sistema aborda requisitos complexos de negócio incluindo controle de estoque, vendas, usuários e auditoria, demonstrando a viabilidade da integração entre bancos relacionais e NoSQL em um contexto empresarial real.

Palavras-chave: Sistema Híbrido, MySQL, MongoDB, Node.js, React, Gestão de Livraria

SUMÁRIO

1. INTRODUÇÃO

A gestão de livrarias envolve operações complexas que demandam tanto a consistência transacional de bancos relacionais quanto a flexibilidade e performance de bancos NoSQL. Este projeto desenvolveu uma solução integrada que aproveita os pontos fortes de ambas as tecnologias, criando um sistema robusto e escalável para administração de uma livraria.

2. Objetivos

- Desenvolver um Sistema completo de gestão para livrarias.
- Implementar arquitetura híbrida MySQL + MongoDB.
- Demonstrar a integração eficiente entre bancos relacionais e NoSQL.
- Fornecer controle granular de acesso e auditoria completa.
- Otimizar performance através de cache inteligente.

3. METODOLOGIA

O desenvolvimento seguiu uma abordagem ágil, com implementação iterativa das funcionalidades. A arquitetura foi planejada para separar claramente as responsabilidades entre os bancos de dados, utilizando MySQL para operações transacionais críticas e MongoDB para dados de suporte e análise.

4. DESCRIÇÃO DO SISTEMA

4.1 Funcionalidades

- Gestão de Catálogo: Livros, autores, editoras e categorias.
- Controle de Estoque: Gestão em tempo real de quantidade disponível.
- Vendas e Pedidos: Processamento transacional completo.
- Clientes e Usuários: Cadastro e gestão de acessos.
- Auditoria: Log completo de todas as operações.
- Relatórios: Estatísticas e análise em tempo real.
- Controle de Acesso: Sistema baseado em grupos e permissões.

4.2 Tecnologias Utilizadas

Frontend

- React: Biblioteca para interface do usuário.
- CSS3: Estilização e flexibilidade.

Backend

- Node.js: Runtime JavaScript server-side.

- Express.js: Framework web para APIs RESTful.
- JWT: Autenticação baseada em tokens.
- bcrypt: Criptografia de senhas.

Banco de Dados Relacional

- MySQL 8.0: SGBD relacional principal.
- Banco NoSQL
- MongoDB: Banco de dados documental.

4.3 Justificativa de Cada Escolha Tecnológica

MySQL

Justificativa: Escolhido como banco principal devido à:

- Consistência ACID: Essencial para operações financeiras e de estoque.
- Integridade Referencial: Garante relacionamentos consistentes entre entidades.
- Transações Complexas: Suporte a operações multi-tabela em pedidos.
- SQL Maduro: Amplo suporte a views, procedures e triggers.

MongoDB

Justificativa: Implementado para funcionalidades específicas:

- Flexibilidade Schema-less: Ideal para logs de auditoria com estruturas variáveis.
- Performance de Leitura: Cache de consultas frequentes.
- Escalabilidade Horizontal: Preparado para crescimento de dados de log.
- Agregações Nativas: Perfeito para análise e estatísticas.

Node.js + Express

Justificativa:

- JavaScript Full-Stack: Uniformidade de linguagem.
- Alta Concorrência: Modelo non-blocking I/O.
- Ecossistema Rico: Ampla variedade de pacotes npm.

React

Justificativa:

- Composição: Reutilização e manutenção de código.

- Virtual DOM: Performance otimizada de interface.
- Estado Previsível: Gestão consistente do estado da aplicação.

5. Modelagem do Banco de Dados

5.1 Diagrama Entidade-Relacionamento (DER)

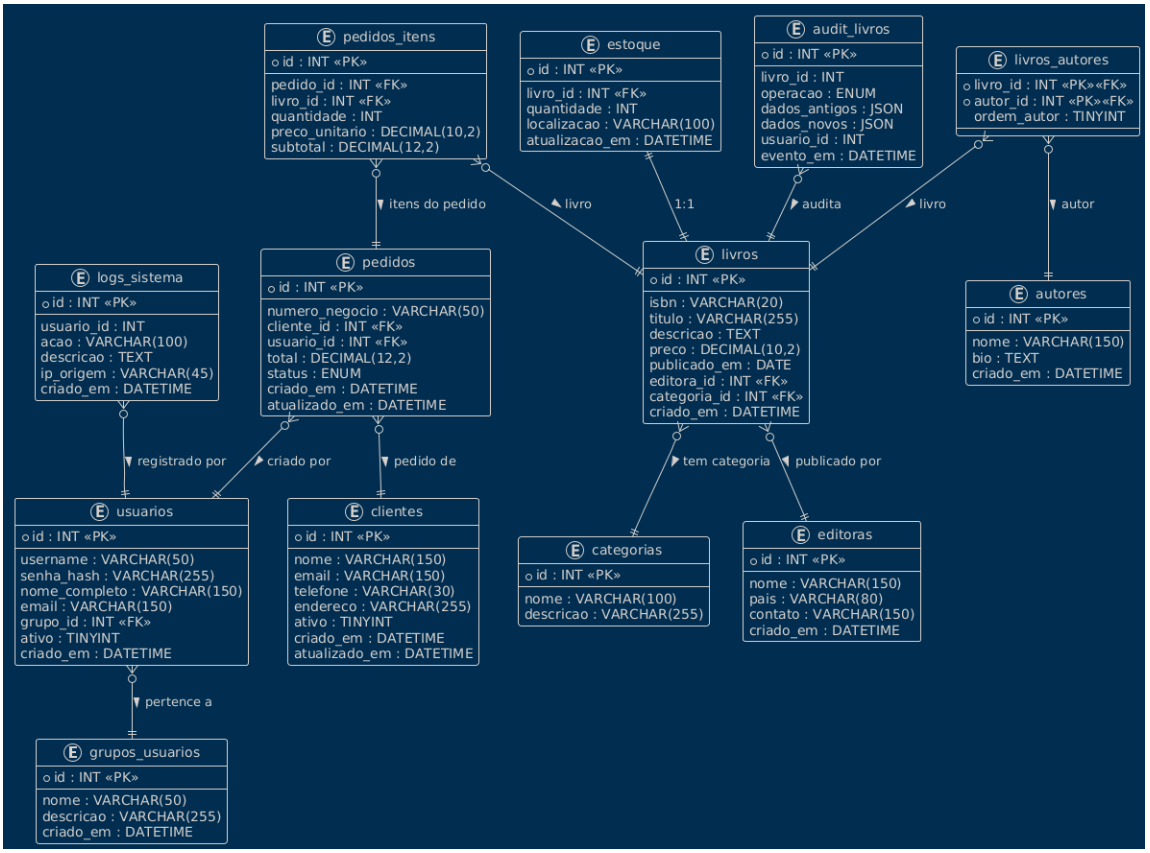

Figura 1 Fonte: Elaboração própria (2025).

5.2 Explicação das Entidades e Relacionamentos

Entidades Principais

grupos_usuarios ↔ usuarios (1:N)

- Relação: Um grupo contém múltiplos usuários.
- Cardinalidade: 1 para muitos.
- Justificativa: Controle hierárquico de acesso.

clientes ↔ pedidos (1:N)

- Relação: Um cliente faz múltiplos pedidos.
- Cardinalidade: 1 para muitos.
- Justificativa: Histórico completo de compras por cliente.

livros ↔ pedidos_itens (1:N)

- Relação: Um livro pode aparecer em múltiplos itens de pedido.
- Cardinalidade: 1 para muitos.
- Justificativa: Rastreabilidade de vendas por livro.

livros ↔ autores (N:M via livros_autores)

- Relação: Muitos-para-muitos entre livros e autores.
- Justificativa: Um livro pode ter múltiplos autores, um autor pode ter múltiplos livros.

6. Justificativa para Elementos Avançados

Índices

```
CREATE INDEX idx_livro_titulo ON livros(titulo);
CREATE INDEX idx_livro_isbn ON livros(isbn);
CREATE INDEX idx_pedidos_cliente ON pedidos(cliente_id);
```

Figura 2 Fonte: Elaboração própria (2025).

Imagem2

Justificativa: Otimização de consultas frequentes por título, ISBN e histórico de cliente.

Triggers

```
CREATE TRIGGER trg_audit_livros_after_update
AFTER UPDATE ON livros FOR EACH ROW
BEGIN
    INSERT INTO audit_livros (...) VALUES (...);
END
```

Figura 3 Fonte: Elaboração própria (2025).

Imagem3

Justificativa: Auditoria automática de alterações críticas no catálogo.

Views

```
CREATE VIEW vw_livros_disponiveis AS
SELECT l.id, l.titulo, l.preco, e.nome as editora, est.quantidade
FROM livros l
JOIN editoras e ON l.editora_id = e.id
JOIN estoque est ON l.id = est.livro_id;
```

Figura 4 Fonte: Elaboração própria (2025).

Imagem4

Justificativa: Simplificação de consultas complexas frequentes.

Stored Procedures

```
CREATE PROCEDURE sp_criar_pedido(...)
BEGIN
    -- Lógica transacional complexa
END
```

Figura 5 Fonte: Elaboração própria (2025).

Imagem 5

Justificativa: Encapsulamento de lógica de negócio complexa para pedidos.

6.1 Descrição dos Usuários e Grupos

Grupos Definidos:

ADMIN: Acesso total ao sistema.

ATENDIMENTO: Gestão de pedidos, clientes e estoque.

LEITURA: Apenas consulta de dados.

6.2 Regras de Acesso

```
// Middleware de autorização
function authorize(requiredGroups = []) {
    return (req, res, next) => {
        if (!requiredGroups.includes(req.user.grupo)) {
            return res.status(403).json({ error: "Forbidden" });
        }
        next();
    };
}

// Exemplo de uso
router.post('/livros', authenticate, authorize(['ADMIN','ATENDIMENTO']), livrosContro
ller.create);
```

Figura 6 Fonte: Elaboração própria (2025).

Imagem6

Justificativa: Controle granular baseado em grupos, seguindo princípio do menor privilégio.

7. Uso do Banco NoSQL

7.1 Explicação Técnica do MongoDB

Estrutura de Documentos:

```
class AuditService {
    static async logAction(collectionName, documentId, operation, oldData, newData, u
serId) {
        await AuditLog.create({
            collectionName,
            documentId,
            operation,
            oldData,  // Schema-flexible
            newData,  // Estrutura variável
            userId,
            timestamp: new Date()
        });
    }
}
```

Figura 7 Fonte: Elaboração própria (2025).

Imagem7

7.2 Justificativa da Aplicação no Sistema

Casos de Uso do MongoDB:

1. Auditoria e Logs

```
class AuditService {
    static async logAction(collectionName, documentId, operation, oldData, newData, u
serId) {
        await AuditLog.create({
            collectionName,
            documentId,
            operation,
            oldData,  // Schema-flexible
            newData,  // Estrutura variável
            userId,
            timestamp: new Date()
        });
    }
}
```

Figura 8 Fonte: Elaboração própria (2025).

Imagem8

Justificativa: Estrutura de logs imprevisível beneficia-se do schema-less do MongoDB.

2. Cache de Performance

```
// Cache de Livros Populares
class CacheService {
    static async getTopSellingBooks(limit = 10) {
        return await BookCache.find()
            .sort({ salesCount: -1 })
            .limit(limit)
            .lean();
    }
}
```

Figura 9 Fonte: Elaboração própria (2025).

Imagem9

Justificativa: Consultas frequentes de análises com performance otimizada.

3. Sessões de Usuário

```
// Gerenciamento de Sessões
userSessionSchema.index({ expiresAt: 1 }, { expireAfterSeconds: 0 });
```

Figura 10 Fonte: Elaboração própria (2025).

Imagem10

Justificativa: TTL automático para sessões, nativo no MongoDB.

Vantagens da Abordagem Híbrida:

- MySQL: Garante consistência nas operações críticas (vendas, estoque).
- MongoDB: Oferece flexibilidade para dados de suporte (logs, cache).
- Performance: Cache inteligente reduz carga no MySQL.
- Escalabilidade: Cada banco escala conforme sua natureza de uso.

8. Conclusão

O sistema desenvolvido demonstra com sucesso a viabilidade da integração entre bancos relacionais e NoSQL em um contexto empresarial real. A arquitetura híbrida MySQL+MongoDB mostrou-se eficiente ao:

- **Manter consistência** nas operações críticas através do MySQL.
- **Oferecer flexibilidade** para funcionalidades auxiliares via MongoDB.
- **Otimizar performance** através de cache inteligente.

- **Garantir rastreabilidade** com auditoria completa.

A escolha criteriosa de quando utilizar cada tecnologia permitiu criar um sistema robusto, escalável e com manutenção aprimorada. Este projeto serve como referência para implementações similares que demandam tanto a confiabilidade de bancos relacionais quanto a agilidade de soluções NoSQL.

9. Referências

1. MYSQL. MySQL 8.0 Reference Manual. Oracle Corporation, 2024.
2. MONGODB. MongoDB Documentation. MongoDB, Inc., 2024.
3. NODE.JS. Node.js Documentation. Node.js Foundation, 2024.
4. REACT. React Documentation. Facebook Open Source, 2024.
5. FOWLER, M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.
6. SADALAGE, P. J.; FOWLER, M. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley, 2012.