DOCUMENTAÇÃO SISTEMA DE GERENCIAMENTO BANCÁRIO

BANKING MANAGEMENT SYSTEM DOCUMENTATION

João Pedro de Oliveira Lima*
Manoel Pedro Prado Sá Teles*
Leydson Douglas Góes de Araújo Praseres*
Luís Fernando Ventura Ferreira*
Lucas Costa Silva*
Gustavo Nascimento Martins*

RESUMO

O objetivo deste artigo é apresentar o desenvolvimento de um Sistema de Gerenciamento Financeiro (SGF) concebido para otimizar e automatizar operações bancárias essenciais, com foco na alta disponibilidade e integridade de dados. A arquitetura do sistema foi delineada para suportar a lógica complexa e pesada das transações, que constitui o núcleo operacional da solução. Adota-se uma metodologia de desenvolvimento ágil para a modelagem, implementação e validação de módulos críticos. O SGF permite o cadastro e a gestão de usuários com perfis de cliente e gerência, facilitando a criação de contas e o gerenciamento de chaves PIX para transações instantâneas, além da geração de boletos. Para garantir a segurança e a consistência, o sistema realiza verificações automáticas e em tempo real de saldo e de existência de conta antes de processar qualquer movimentação. Os resultados demonstram a viabilidade de uma plataforma transacional segura e eficiente, provendo uma solução robusta para a administração de recursos financeiros.

Palavras-chave: Gerenciamento Financeiro; Transações Bancárias; PIX; Integridade de Dados; Arquitetura de Sistemas.

ABSTRACT

The purpose of this article is to present the development of a Financial Management System (FMS) designed to optimize and automate essential banking operations, focusing on high availability and data integrity. The system architecture is outlined to support the complex and heavy transaction logic, which constitutes the operational core of the solution. An agile development methodology is adopted for the modeling, implementation, and validation of critical modules. The FMS allows the registration and management of users with client and management profiles, facilitating account creation and the management of PIX keys for instant transactions, in addition to boleto generation. To ensure security and consistency, the system performs automatic and real-time verifications of balance and account existence before processing any movement. The results demonstrate the feasibility of a secure and efficient transactional platform, providing a robust solution for the administration of financial resources.

Keywords: Financial Management; Banking Transactions; PIX; Data Integrity; System Architecture.

1 INTRODUÇÃO

De acordo com padrões de estrutura de software e boas práticas recomendadas, foi desenvolvido uma aplicação web que representa uma fração da estrutura de um sistema de gerenciamento bancário, tema escolhido por ser complexo o suficiente para conter grande parte das ferramentas utilizadas dentro do DBMS (*Data Base Management System*) escolhido, como também sua integração em um artefato de software completo, contendo *front-end*, *back-end* e banco de dados.

2 OBJETIVOS

O projeto possui como Objetivo Geral desenvolver uma plataforma de gerenciamento financeiro capaz de processar transações de forma segura e eficiente, integrando tecnologias frontend, backend e bancos de dados relacionais e NoSQL.

Os Objetivos Específicos são:

- Modelar e implementar um banco de dados transacional robusto que suporte o cadastro de usuários (clientes e gerentes) e a criação de contas.
- Desenvolver a lógica de backend para o processamento de transações, incluindo transferências (PIX) e geração de boletos.
- Demonstrar a aplicação de mecanismos avançados do SGBD, como índices, triggers e procedimentos, para garantir a integridade e o desempenho do sistema.
- Justificar de forma clara e técnica a escolha de cada tecnologia e ferramenta utilizada na arquitetura da solução.

3 METODOLOGIA

O desenvolvimento do Sistema de Gerenciamento Financeiro (SGF) adotou o paradigma de **Desenvolvimento Ágil**, visando entregas incrementais e validação contínua, o que foi essencial para lidar com a complexidade do módulo transacional. A arquitetura da solução foi projetada de forma modular, separando as camadas *frontend*, *backend* e de persistência. A fase de *design* do banco de dados utilizou a **Modelagem Entidade-Relacionamento (MER)** para garantir a integridade dos dados, com foco na implementação de otimizações do SGBD que assegurassem a robustez e o desempenho exigidos por um sistema financeiro.

4 DESCRIÇÃO DO SISTEMA

A arquitetura do Sistema de Gerenciamento Financeiro (SGF) adota uma abordagem *Full-Stack* modular, combinando diferentes tecnologias para otimizar desempenho, manutenibilidade e, acima de tudo, a integridade transacional do sistema.

Camada	**Tecnologia**	**Justificativa Essencial**
Frontend	HTML, CSS, JavaScript (Puro)	Escolha por manter o código enxuto e de fácil manutenção, garantindo controle total sobre o desempenho e a responsabilidade única de cada arquivo.
Backend	Java com	Utilização de uma stack robusta e consolidada no

Camada	Tecnologia	Justificativa Essencial
	Framework Spring	mercado financeiro. O Java é escalável e o Spring oferece facilidade em mapeamento de dados (ORM).
SGBD Relacional	MySQL	Banco de dados transacional escolhido por garantir Atomicidade, Consistência, Isolamento e Durabilidade, Permite a aplicação de Triggers e Stored Procedures.
SGBD NoSQL	MongoDB Atlas	Armazena documentos não-estruturados (PDFs de boletos), reduzindo a carga do backend e do banco relacional ao evitar a regeneração de documentos já criados.

Tabela 01 – Estrutura do Sistema – Fonte: autor

5 MODELAGEM DO BANCO DE DADOS

Esta seção detalha a estrutura da persistência de dados, crucial para garantir a integridade e a performance das operações financeiras do sistema.

5.1 Diagrama Entidade Relacionamento (DER)

O Diagrama Entidade-Relacionamento (DER) representa visualmente a estrutura conceitual do banco de dados, estabelecendo as entidades e as relações que governam o fluxo de dados.

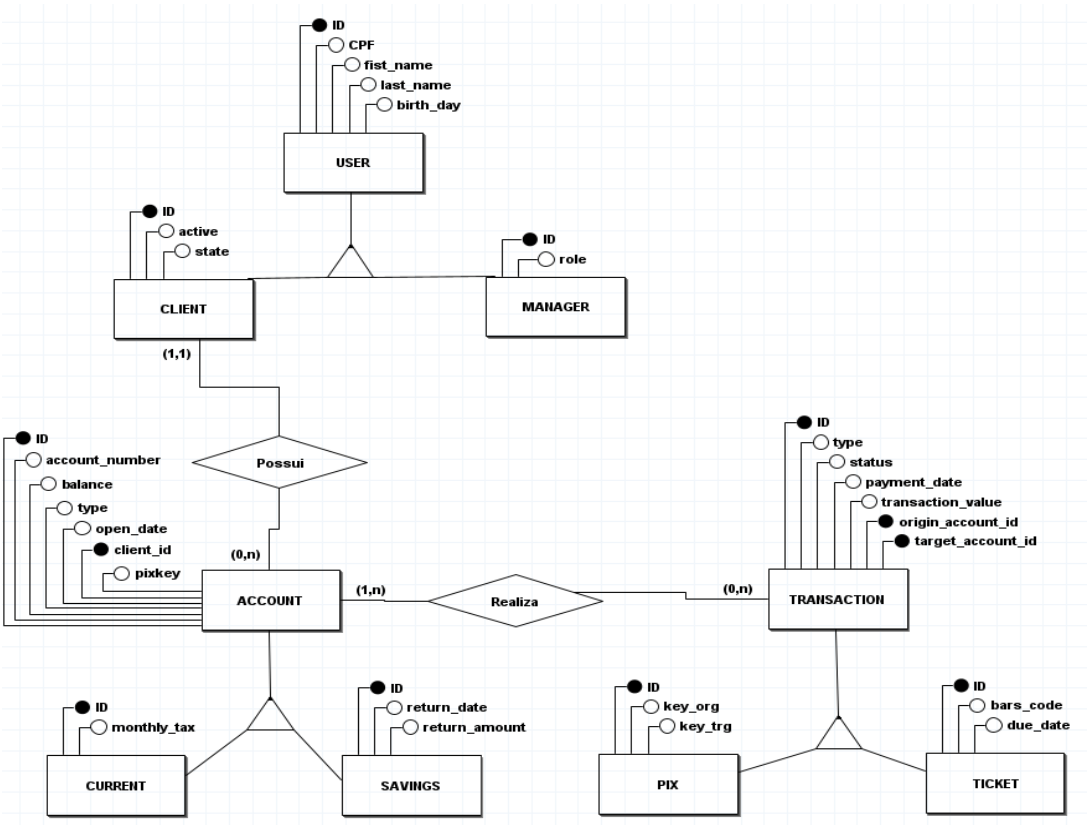

Imagem 01 – Modelo Conceitual de Dados – Fonte: autor

5.2 Explicação das Entidades e Relacionamentos

A estrutura utiliza o conceito de herança (generalização/especialização) para diferenciar os tipos de usuários e contas, conforme o DER (Imagem 01).

Entidade Principal	Descrição	Relacionamento Chave
USER	Dados básicos de todos os usuários (CPF, nome, data de nascimento).	1:1 CLIENT ou MANAGER.
CLIENT MANAGER	Especializações do USER. CLIENT possui o status de atividade e estado.	1:N de CLIENT para ACCOUNT.
USER_GROUP	Tabela auxiliar que armazena o tipo de grupo (1 para gerente, 2 para cliente) ao qual o usuário pertence.	1:1 com USER (via ID).
ACCOUNT	Entidade central que registra o saldo, o número da conta, chaves pix e o tipo.	1:N de ACCOUNT para TRANSACTION. 1:1 de ACCOUNT para PIXKEY.
CURRENT SAVINGS	Especializações de ACCOUNT. Armazenam informações específicas, como taxa mensal ou valor de rendimento.	N/A
PIXKEY	Armazena a chave PIX (email, telefone ou chave aleatória).	1:1 com ACCOUNT.
TRANSACTION	Registra todos os movimentos (PIX, Boleto). Contém o valor, o *status* e as chaves de origem e destino.	1:1 com PIX ou TICKET.
PIX / TICKET	Detalham as transações específicas. PIX armazena as chaves usadas. TICKET armazena o código de barras e a data de vencimento.	1:1 com TRANSACTION.

Tabela 02 – Entidades e Relacionamentos do Banco de Dados – Fonte: autor

5.3 Uso de Indexes, Triggers, Views e Functions/Procedures

Uso e justificativa para cada elemento entre índices, gatilhos, tabelas virtuais e funções.

5.3.1 Indexes

- Recurso: idx_transaction_type e idx_transaction_status na tabela tb_transaction.
- Justificativa: A tabela tb_transaction é a que mais cresce e é mais consultada. Criar índices nos campos type e status acelera drasticamente as consultas de relatórios e a

busca de informações de pagamento, otimizando a performance do sistema, especialmente sob alta carga transacional.

5.3.2 Triggers

- Recurso: insert_group_client, insert_group_manager, e remove_pixkeys_account.
- Justificativa: As Triggers de grupo (insert_group_client e insert_group_manager) automatizam a gestão de permissões. Ao inserir um novo cliente, o sistema garante automaticamente que ele receba as permissões de crud_user na tabela tb_userGroup, mantendo a consistência do controle de acesso.
  A Trigger remove_pixkeys_account é essencial para a conformidade e segurança. Quando um cliente é desativado, esta Trigger remove automaticamente as chaves PIX associadas às suas contas, evitando o uso indevido de canais de pagamento em contas inativas.

5.3.3 Views

- Recurso: vw_client_account_summary e vw_transaction_report.
- Justificativa: vw_client_account_summary - Simplificação e segurança, a pesquisa se torna mais fácil e para usuários com menos permissões vejam apenas os dados necessários. vw_transaction_report: Clareza para relatórios, unindo todas as transações, com os dados específicos de pix e boletos.

5.3.4 Functions/Procedures

- Recurso: executePixPayment e executeTicketPayment.
- Justificativa: Ao encapsular toda a lógica de débito, crédito e registro da transação dentro de uma única rotina de banco de dados, o uso de START TRANSACTION e COMMIT/ROLLBACK garante que as operações ocorram de forma atômica. Se o saldo for insuficiente ou a chave PIX não for encontrada, o ROLLBACK é acionado, prevenindo inconsistências de saldo.

6 CONTROLE DE ACESSO

O controle de acesso foi feito através da criação de usuários específicos para funcionalidades distintas, melhorando a segurança e garantindo a integridade do banco de dados.

6.1 Acesso de Sistema

- User: system_access - Usuário de conexão do Backend.
- Role: crud_user - Permissão CRUD nas tabelas transacionais e a permissão EXECUTE nas *Stored Procedures* (para transações).

6.2 Acesso de Sistema

- User: employee - Usuário de Gerência/Administração.
- Role: admin - Permissões de CRUD e permissões estruturais como CREATE, ALTER, INDEX, CREATE VIEW. Necessário para tarefas de administração e manutenção do banco.

7 USO DO BANCO NoSQL

O projeto utiliza o conceito de Persistência Poliglota ao integrar o MongoDB Atlas, um banco de dados NoSQL orientado a documentos, em conjunto com o SGBD relacional. Tecnicamente, o MongoDB armazena dados no formato BSON (JSON Binário) e oferece um

esquema flexível, ideal para gerenciar dados cuja estrutura não é rígida. Sua aplicação específica no sistema é servir como armazenamento de *cache* para documentos não-estruturados, como os PDFs de boletos gerados. Esta escolha é estratégica, pois evita a regeneração do documento a cada nova solicitação por parte do usuário, o que reduz significativamente a carga de processamento no *backend* e do banco de dados transacional, otimizando assim o tempo de resposta e a performance geral do sistema.

8 CONCLUSÃO

O desenvolvimento do Sistema de Gerenciamento Financeiro (SGF) cumpriu o objetivo de criar uma plataforma robusta e escalável, demonstrando a aplicação de boas práticas de *software* e a utilização estratégica de um modelo de persistência híbrida. A implementação de Stored Procedures no MySQL garantiu a integridade das operações críticas (PIX e Boleto), enquanto o uso do MongoDB otimizou a entrega de documentos. O Controle de Acesso assegurou a segregação de permissões entre o sistema e os usuários administradores, resultando em uma solução segura e eficiente, pronta para ser escalada.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6022**: informação e documentação - artigo em publicação periódica técnica e/ou científica - apresentação. Rio de Janeiro: ABNT, 2018a.

MYSQL. **MySQL 8.4 Reference Manual. Oracle, 2025**. Disponível em: https://dev.mysql.com/doc/en/. Acesso em: [02 nov. 2025].

SPRING. **Spring Boot Reference Documentation. Pivotal, [S.d.]**. Disponível em: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/. Acesso em: [02 nov. 2025]. (Nota: O [S.d.] significa Sem Data, usado quando a data de publicação não está clara ou é dinâmica, como em documentações online).

ORACLE. **CRUD REST utilizando Spring Boot 2, Hibernate, JPA, e MySQL**. Autor: Loiane Groner. [S.l.]: Oracle Brasil, fev. 2019. Disponível em: https://www.oracle.com/br/technical-resources/articles/dsl/crud-rest-sb2-hibernate.html. Acesso em: [05 nov. 2025].

EMANUEL, Vitor. **API REST com Spring Boot, Spring Data JPA, Maven e PostgreSQL**. In: Medium. 9 jan. 2021. Disponível em: https://vitoremanueldev.medium.com/api-rest-com-spring-boot-2-spring-framework-hibernate-jpa-maven-e-postgresql-b81b5c7952a7. Acesso em: [09 nov. 2025].